UNIVERSIDAD JAUME I
DPTO. DE INGENIERÍA Y CIENCIA DE LOS COMPUTADORES

Técnicas de agrupamiento bilingüe aplicadas a la inferencia de traductores

Tesis Doctoral
Presentada por Sergio Barrachina Mir
Dirigida por el Doctor D. Juan Miguel Vilar Torres

Castellón, marzo de 2003

Técnicas de agrupamiento bilingüe aplicadas a la inferencia de traductores

Sergio Barrachina Mir

Trabajo realizado bajo la dirección del Doctor
D. Juan Miguel Vilar Torres
y presentado en la Universidad Jaume I
para optar al grado de Doctor en Ingeniería Informática

Castellón, marzo de 2003

Este trabajo ha sido parcialmente realizado dentro de los proyectos TransType 2 (proyecto IST-2001-32091), subvencionado por la Unión Europea; SIEsTA (proyecto P1·1B2002-15), subvencionado por BanCaja; NeuroTrad (proyecto P1A99-10), subvencionado por BanCaja; EuTrans (proyecto ESPRIT n. 30268), subvencionado por la Unión Europea y ExTra (TIC97-0745-C02-01), subvencionado por la Comisión Interministerial de Ciencia y Tecnología.

A mi amor, Rosa.

A mis padres, Salvador y Lupe.

A mis hermanas, Katya y Beatriz.

A mis sobrinitas, Ana, Olga y Sara.

No podría ser más afortunado.

Resumen

Presentamos un método de agrupamiento bilingüe que puede utilizarse para la mejora de sistemas de traducción automática basados en ejemplos. Este método de agrupamiento es una extensión del caso monolingüe y es mejorado mediante la detección automática de colocaciones —secuencias de palabras que se traducen como una unidad—. Asimismo, presentamos una técnica para la integración de este agrupamiento con un sistema de traducción basado en ejemplos, el modelo de transductores subsecuenciales, y los resultados de traducción obtenidos.

Los asuntos que tratamos en esta tesis son:

- Agrupamiento monolingüe.
- Agrupamiento bilingüe.
- Integración del agrupamiento bilingüe en transductores subsecuenciales.
- Detección automática de colocaciones.
- Mejora del agrupamiento bilingüe mediante colocaciones.
- Experimentación con las tareas de traducción EuTrans I y EuTrans II.

II

Prólogo

En esta tesis nos planteamos la mejora en el aprendizaje de sistemas de traducción basados en ejemplos (STBEs). Uno de los métodos empleados con éxito para la mejora de estos sistemas consiste en la agrupación de determinadas palabras de los idiomas objeto de la traducción, en un conjunto de categorías. Sin embargo, puesto que la definición de estas categorías es realizada por expertos humanos, utilizando este método se reduce una de las ventajas inherentes a los sistemas basados en ejemplos: su adaptabilidad. Un STBE que no utilice categorías puede adaptarse fácilmente a un nuevo dominio de traducción, o a un nuevo par de lenguas; prácticamente es suficiente con disponer de un conjunto adecuado de ejemplos para el nuevo dominio o los nuevos lenguajes. Utilizando categorías definidas de forma manual, para adaptar un STBE sería necesario, además, estudiar previamente qué categorías son apropiadas para cada nueva tarea de traducción y qué palabras de ambos idiomas forman parte de cada una de ellas.

Proponemos un método automático de agrupamiento bilingüe que pretende suplir a la categorización manual como método para la mejora de los STBEs. Este método obtiene de forma automática un agrupamiento bilingüe, utilizando únicamente un conjunto de ejemplos de frases en un idioma y su correspondiente traducción. La misma información que se requiere para el entrenamiento de un STBE sin categorías.

La técnica de agrupamiento bilingüe propuesta se basa en la extensión de métodos de agrupamiento monolingüe. Presentamos también en esta tesis tres métodos de agrupamiento monolingüe. Los dos primeros son métodos iterativos de agrupamiento y el tercero es un método incremental. Los iterativos parten de una distribución inicial de palabras en el número de clases deseado y mueve iterativamente cada palabra a cada una de las clases disponibles. Para evaluar la idoneidad de cada movimiento utilizan una función objetivo: la información mutua entre clases adyacentes. En el primero de ellos, esta función objetivo se calcula directamente a partir del corpus de entrenamiento. En el segundo de los métodos se calcula utilizando el método de *dejar uno fuera* para simular eventos no vistos. Por último, el método incremental genera el agrupamiento comenzando

con una clase e incrementa el número de clases hasta llegar al número de clases deseado. Para cada número distinto de clases realiza un agrupamiento iterativo; pudiendo utilizar a cualquiera de los otros dos métodos para este fin.

Además, la forma en la que extendemos los algoritmos monolingües al caso bilingüe, hace conveniente la identificación automática de secuencias de palabras que se comporten, desde el punto de vista de la traducción, como una unidad. Hemos denominado *colocaciones* a dichas secuencias. Presentamos también un algoritmo para la identificación automática de colocaciones.

Consideramos que estos algoritmos, los de agrupamiento monolingüe y el de identificación de colocaciones, que en esta tesis se emplean para la obtención de agrupamientos bilingües, pueden tener interés por sí mismos y ser aplicables a otras tareas de tratamiento de lenguaje natural.

Por último, hemos realizado un conjunto de experimentos de traducción con el fin de comprobar la idoneidad de las técnicas propuestas. En estos experimentos hemos utilizado como STBE el modelo de transductores subsecuenciales (SSTs). Para la inferencia de SSTs hemos utilizado el algoritmo OMEGA. Los experimentos se han llevado a cabo con dos tareas de traducción: una relativamente sencilla (EuTrans I) y otra más compleja (EuTrans II). Los excelentes resultados obtenidos con la primera de ellas nos induce a creer en la validez de las técnicas presentadas. Desgraciadamente, no hemos obtenido similares resultados con la segunda tarea. Creemos, sin embargo, que esto es debido a que el número de ejemplos de entrenamiento es insuficiente para la complejidad de esta última y que habríamos obtenido mejores resultados de haber dispuesto de más ejemplos. Esta creencia se ve reforzada al constatar que otros métodos de agrupamiento monolingüe tampoco obtienen resultados positivos. No obstante, pretendemos realizar más experimentos sobre otros corpora de igual o mayor complejidad, pero con más ejemplos de entrenamiento, para contrastar este último punto; además de para mejorar las técnicas propuestas.

El resto de la tesis está organizada como sigue:

- Capítulo 1. Introducimos la problemática de la traducción, los principales paradigmas de traducción automática, los sistemas de traducción basados en ejemplos y los objetivos de la tesis.
- Capítulo 2. Dedicamos este capítulo a la introducción de la notación básica y a una serie de conceptos generales comunes al resto de capítulos.
- Capítulo 3. Presentamos varias aproximaciones al agrupamiento de palabras en un idioma empleando métodos estadísticos. Proponemos tres métodos que serán utilizados como base del agrupamiento bilingüe.
- Capítulo 4. Exponemos la técnica propuesta para la realización de agrupamientos bilingües que sean útiles en el contexto de la traducción automática. Describimos la integración de dicho agrupamiento en un STBE durante la fase de aprendizaje y las modificaciones necesarias para llevar a cabo la fase de traducción.

- Capítulo 5. En este capítulo proponemos un método para la identificación automática de colocaciones y mostramos cómo integrar las colocaciones en el método de agrupamiento bilingüe propuesto en el capítulo anterior.
- Capítulo 6. Mostramos aquí los experimentos realizados en los que comparamos los métodos propuestos de agrupamiento monolingüe con otros métodos existentes. También hemos realizado experimentos de traducción en el que comparamos los resultados cuando se utilizan o no agrupamientos bilingües y/o colocaciones.
- Capítulo 7. Exponemos las conclusiones de esta tesis y los desarrollos futuros que nos gustaría llevar a cabo.
- Apéndice A. Presentamos los agrupamientos bilingües obtenidos mediante las técnicas propuestas para la tarea de traducción EuTrans I, así como las colocaciones identificadas de forma automática.
- Apéndice B. Detallamos en este apéndice los resultados numéricos correspondientes a los experimentos mostrados en el capítulo 6.

Agradecimientos

La escritura de esta tesis no hubiera sido posible sin la colaboración de muchas personas. Quisiera mostrar aquí mi agradecimiento a algunas de ellas. En primer lugar, me gustaría dar las gracias a Juan Miguel Vilar Torres por su ayuda en mi formación, que ha ido mucho más allá del campo de esta tesis, y por su continuo apoyo en los sinsabores de la investigación. Además de por las incontables revisiones de borradores de esta tesis. Los errores que hayan quedado son míos.

A Fede Prat Villar por su desinteresada colaboración siempre que la he necesitado y por su estímulo y sugerencias en la reimplementación del programa *e-cluster*. A Andrés Marzal Varó por reforzar mi vocación universitaria y por haberme dado la oportunidad de comenzar a trabajar en este campo de investigación. A Toni Castellanos López y a Germán León Navarro por su constante apoyo; muy especialmente al principio de mi andadura por esta Universidad.

A Merche Marqués Andrés, a Rafael Mayo Gual y a Asunción Castaño Álvarez por haber accedido a revisar partes de esta tesis y, sobre todo, por sus acertadas sugerencias. Me gustaría agradecer también a Rafael Mayo Gual su lema +T-T. Y a Maribel Castillo Catalán, el que no haya dejado que Rafa me agobiara más de la cuenta.

Finalmente, a mis compañeros del grupo ÁCRATA y a los de los Dptos. de Ingeniería y Ciencia de los Computadores y de Lenguajes y Sistemas Informáticos de la Universitat Jaume I por toda la ayuda prestada.

Índice general

Índice de figuras

Índice de cuadros

Listado de algorítmos

Capítulo 1

Introducción

1.1. Introducción

Desde la aparición de los computadores el hombre ha perseguido la construcción de sistemas de traducción automática. Sistemas capaces de, a partir de un texto en un idioma, generar un texto que exprese en otro idioma el significado del primero. La utilidad de un sistema de este tipo está fuera de toda duda. Es fácil pensar en aplicaciones a nivel personal, empresarial o institucional.

Mediante un sistema de traducción automática, cualquiera podría acceder a información escrita que no estuviera disponible en su idioma nativo. Esta aplicación es aún más apremiante con la proliferación del uso de Internet y el fácil acceso a cantidades ingentes de información. Podríamos difundir un mismo mensaje en varios idiomas simultáneamente. Sería posible la comunicación fluida entre un grupo de personas sin importar el idioma que hablaran. Podríamos viajar a cualquier lugar del mundo sin que el idioma fuera un gran impedimento.

Las repercusiones económicas que tendría la traducción automática son también sumamente interesantes. Por ejemplo, en sistemas de reserva relacionados con viajes y turismo donde a menudo la capacidad de comunicación en distintas lenguas puede marcar la diferencia. Otro ejemplo se daría en el marco de la Unión Europea, donde la obligación de traducir las etiquetas o manuales de los productos supone un serio obstáculo a aquellas empresas que aspiran a distribuir sus productos en la zona comunitaria.

De hecho, cuanto mayor sea el volumen de información que se debe generar en varios idiomas, cuanto mayor sea el número de idiomas y con cuanta más frecuencia dicha información sea susceptible de ser modificada, tanto más interesante resulta el disponer de un sistema que automatice el proceso de traducción.

En esta situación se encuentran un gran número de empresas multinacionales que generan documentación para y desde sus distintas sucursales en diversos países; los consorcios multinacionales y los organismos oficiales como el Parlamento Europeo que debe proporcionar gran parte de su información en, por el momento, 11 idiomas.

Si éstas son sólo algunas de las posibles aplicaciones, con grandes repercusiones económicas algunas de ellas, ¿cómo es que aún no se dispone de sistemas de traducción automática que las lleven a cabo? La respuesta está relacionada con la gran complejidad del lenguaje humano y los problemas que presenta desde el punto de vista de su traducción.

Sin embargo, y aunque en general, la construcción de sistemas capaces de realizar automáticamente traducciones de calidad entre textos cualesquiera de dos idiomas dados no parece un objetivo alcanzable a medio plazo (salvo para pares de idiomas muy similares entre sí), existen aplicaciones más modestas y prácticas en las que un sistema de traducción automática puede ser de gran utilidad. Una de dichas aplicaciones es la ayuda a los traductores profesionales mediante la predicción una a una de las palabras que forman la traducción de un texto. De esta forma, el traductor sólo necesita teclear las correcciones necesarias a la traducción sugerida conforme ésta se va generando. La idea principal en los sistemas de este tipo es la de liberar al traductor profesional de las labores repetitivas de la traducción, dejando que se concentre únicamente en aquellos pasajes más complejos que el sistema de traducción automática no es capaz de traducir correctamente.

Otro objetivo también deseable y que parece alcanzable de forma más inmediata es el que se centra en obtener sistemas de traducción útiles dentro de determinados dominios de discurso. Piénsese, por ejemplo, en el indudable interés de disponer de tales sistemas en dominios tales como la reserva de habitaciones de hotel o la consulta de horarios y características de medios de transporte internacionales. Existen en la actualidad varios sistemas de traducción automática de este tipo que han sido especialmente diseñados para un dominio particular. Por ejemplo, para la traducción de partes meteorológicos entre inglés y francés. Dichos sistemas pueden realizar traducciones automáticas prácticamente intachables. Sin embargo, no pueden utilizarse para otros dominios ni para otros pares de lenguas, más que para aquellos para los que fueron diseñados. El coste que supone el desarrollo de estos sistemas y el coste de su posible adaptación, por ejemplo, para otro par de lenguas, ha hecho que en la práctica haya pocos de estos sistemas en funcionamiento.

Por lo tanto, parece interesante diseñar un sistema de traducción automática para dominios de discurso limitados capaz de adaptarse a diferentes dominios y a distintos pares de idiomas en un plazo breve de tiempo y con bajos costes tanto de diseño como de implementación. Una posible solución consiste en que el sistema de traducción aprenda a traducir a partir de ejemplos. Sin embargo, aparece entonces el problema de que para conseguir un buen sistema de traducción automática es necesario contar con un gran número de ejemplos.

Se ha comprobado que esta situación puede aliviarse mediante la categorización manual de ambos idiomas. El saber que ciertas palabras forman parte de una determinada categoría permite realizar una serie de generalizaciones sobre los ejemplos disponibles de tal forma que se puede suplir hasta cierto punto la escasez de los datos proporcionados.

Sin embargo, la categorización manual, implica que para adaptar un sistema de traducción de este tipo a otro dominio o a otro par de idiomas, es necesario definir las nuevas categorías en el nuevo dominio, o para el nuevo par de idiomas. Sería deseable poder obtener estas categorías de forma automática.

En esta tesis proponemos una serie de algoritmos de agrupamiento bilingüe que agrupan de forma automática grupos de palabras en dos idiomas. El agrupamiento bilingüe propuesto está pensado para la mejora de sistemas de traducción automática basados en ejemplos. Puesto que los algoritmos bilingües presentados son una extensión de algoritmos monolingües, previamente presentamos una serie de algoritmos de agrupamiento monolingüe en los que se basarán los algoritmos bilingües. Además, debido a la forma en la que se extienden los algoritmos monolingües al caso bilingüe, se hace conveniente la identificación de secuencias de palabras que se traducen como una unidad para mejorar la calidad de los agrupamientos obtenidos y proponemos, por tanto, un método para la identificación automática de dichas secuencias a las que llamamos *colocaciones*.

Pese a que la tesis se desarrolla en el contexto de la traducción automática y busca la construcción de mejores sistemas de traducción automática por medio del agrupamiento bilingüe, creemos que los algoritmos presentados de agrupamiento bilingüe, de agrupamiento monolingüe y de identificación de colocaciones pueden ser interesantes por sí mismos y podrían ser aplicados satisfactoriamente a otras tareas relacionadas con el tratamiento del lenguaje natural como el modelado de lenguaje.

El resto del capítulo está organizado como sigue. La siguiente sección describe algunos de los problemas inherentes a la traducción entre idiomas. La sección 1.3 presenta los principales paradigmas de la traducción automática. Y por último, en la sección 1.4 se presentan los objetivos de esta tesis.

1.2. Problemática de la traducción

La primera aproximación para la generación de un sistema de traducción automática surgió en la década de 1950 [Kni97]. Los antecesores de los actuales computadores habían sido capaces de romper los códigos militares. Fueron utilizados con éxito para decodificar mensajes encriptados. En cierta forma podríamos llegar a decir que lo que estos sistemas hacían era *traducir* del *alemán encriptado* al *alemán*. Estas traducciones se basaban en la sustitución y desplazamiento de las letras que formaban los mensajes encriptados. Una vez acabada la segunda guerra mundial, los criptógrafos y los científicos en teoría de la información se preguntaron si el ruso no podría ser visto como inglés encriptado y la

traducción automática del ruso al inglés como un sistema de descifrado.

Como sistema de cifrado, el ruso parecía bastante complejo. Algunas veces una determinada palabra se codificaba de una forma y otras veces de otra. Las palabras también intercambiaban el orden en el que aparecían —transposición en la jerga criptográfica—. Para poder romper códigos de cifrado complejos, siempre era útil interceptar mensajes en sus formas normal y encriptada (también conocidos como texto plano y cifrado, respectivamente). Afortunadamente, existían mensajes de este tipo tanto en ruso como en inglés: por ejemplo, las traducciones de Tolstoy. Sin embargo, los criptógrafos tuvieron que abandonar esta aproximación debido a la gran envergadura del problema. La encriptación de los mensajes alemanes se había realizado utilizando sencillas máquinas de rotores, mientras que la traducción entre dos idiomas es un asunto mucho más complejo.

Algunos de los problemas que presenta la traducción de un idioma a otro son ocasionados por diferencias léxicas, diferencias estructurales, el uso de anáforas, la determinación del rango de los cuantificadores, las preferencias estilísticas, e incluso por las expresiones utilizadas en el lenguaje hablado [Hut99].

Desde el punto de vista léxico se pueden distinguir dos posibles fuentes de problemas: que el lenguaje origen sea ambiguo o que existan diferencias entre los lenguajes que se desean traducir. Un lenguaje ambiguo léxicamente utiliza una misma palabra para expresar varios conceptos diferentes. El problema surge cuando el lenguaje destino sí que utiliza palabras distintas para representar dichos conceptos. Por ejemplo, la palabra *light* en inglés puede tener cualquiera de los siguientes significados (y su equivalente palabra en castellano): *luz*, *encender*, *claro* o *ligero*. Por lo tanto, si quisiéramos traducir al castellano una determinada aparición de la palabra *light* deberíamos saber cuál de sus posibles acepciones está siendo utilizada para generar la traducción correcta. Por otro lado, dos lenguajes presentan diferencias léxicas cuando matices de un determinado concepto no están representados de igual forma en ambos lenguajes. Por ejemplo, las palabra inglesa *know* engloba a las palabras *saber* y *conocer*; y la palabra *river* se utiliza tanto para hablar de *ríos* como de *afluentes*.

En cuanto a los problemas de tipo estructural, éstos pueden ser debidos a ambigüedades en el lenguaje origen o a diferencias en la forma en la que se dicen las cosas en los idiomas que se desea traducir. Un ejemplo de ambigüedad estructural en el idioma origen sería el siguiente: «*Flying planes can be dangerous*». La frase anterior presenta dos interpretaciones distintas dependiendo de si se considera la palabra *flying* como verbo o como adjetivo. En el primer caso, el sentido de la frase sería el siguiente: «*Pilotar aviones puede ser peligroso*». Mientras que si *flying* actúa como adjetivo el sentido sería: «*Los aviones en vuelo pueden ser peligrosos*». Por otro lado, las diferencias estructurales entre idiomas se deben a que existen formas establecidas para expresar ciertas frases en ambos idiomas que no se traducen literalmente. Así, la expresión inglesa «*to be tough as old boots*» cuya traducción literal en castellano podría ser «*ser fuerte como botas viejas*», en realidad se expresa mediante la frase «*ser fuerte como un roble*».

Los pronombres y determinantes, que se emplean en el discurso para sustituir al sustantivo y evitar su continua repetición, presentan un problema de traducción cuando uno de los lenguajes no distingue entre géneros y el otro sí. El traductor automático tiene que averiguar a qué parte del discurso se está haciendo referencia para poder establecer el género de forma correcta. Por ejemplo, en las siguientes frases, *some of them* debe traducirse por *algunos* o *algunas* dependiendo de si hace referencia a *las niñas* o a *los niños*:

- *The girls threw snowballs to the boys and **some of them** were hit.*
  *Las niñas lanzaron bolas de nieve a los niños y **algunos** fueron alcanzados.*

- *The girls threw snowballs to the boys and **some of them** missed.*
  *Las niñas lanzaron bolas de nieve a los niños y **algunas** fallaron.*

Otro problema similar viene dado por la determinación del alcance de los cuantificadores, es decir, a ¿cuántas palabras está haciendo referencia un determinado cuantificador? En función de la respuesta a esta pregunta se puede interpretar una misma frase de diversas formas. Por ejemplo, en la frase: «*No smoking seats are available on domestic flights*», el cuantificador «*no*» puede referirse sólo a *smoking* o a *smoking seats*. En el primer caso, el significado de la frase sería «*hay asientos de no fumadores en los vuelos nacionales*», mientras que en el segundo, el significado de la frase sería «*no hay asientos de fumadores en los vuelos nacionales*».

Otra dificultad añadida, en el caso de pretender que la traducción automática sea equiparable a la que puede generar un traductor humano, es la de conservar las preferencias estilísticas del original en la traducción. Es decir, la traducción no sólo debe ser fiel al contenido del mensaje sino a la forma en la que el autor escogió expresar dicho mensaje. Por ejemplo, la obra francesa «*En busca del tiempo perdido*» de Marcel Proust posee una estructura circular y esta estructura debería ser respetada en la traducción. Una de las formas en la que Proust consigue esta estructura es haciendo que la palabra *tiempo*, que aparece en la primera frase de la novela, sea también la última palabra de la obra. Pretender que un traductor automático sea capaz de *averiguar* que debe utilizar la palabra *tiempo* como última palabra de un documento debido a que cuando empezó a realizar la traducción, miles de palabras atrás, dicha palabra estaba en la primera frase, no parece, desde luego, un propósito realista.

Un ejemplo de traducción aún más compleja en la que hay que ir un paso más allá del hecho de respetar el estilo del autor es en la traducción de poemas. Un traductor humano, cuando traduce un poema, no sólo intenta plasmar el significado del poema en otro idioma, en realidad, crea un nuevo poema en su idioma, de acuerdo a la interpretación que él dé al significado del poema en su conjunto y al estilo del autor original. De hecho, la traducción de un poema es tan dependiente del traductor, que los poemas traducidos se identifican necesariamente por el nombre del autor y el de su traductor. Existiendo, como es natural, diversas traducciones de un mismo poema.

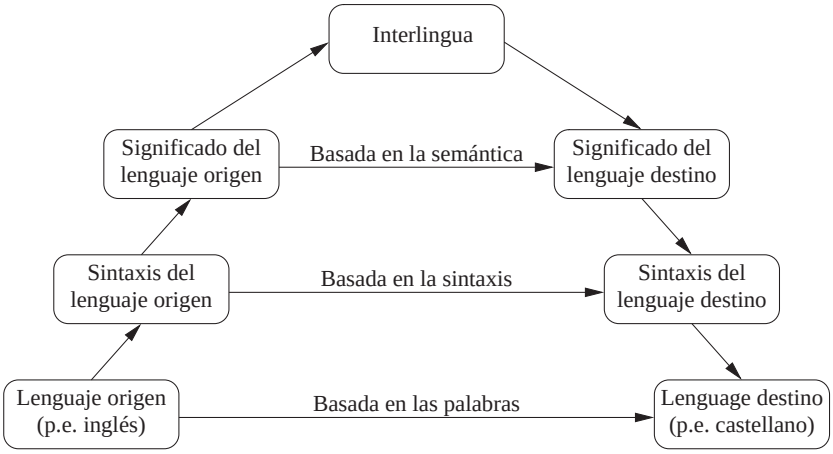


Figura 1.1: Paradigmas de traducción automática.

Para acabar, el discurso hablado también añade una nueva serie de dificultades ya que utiliza formulas de cortesía, contracciones, modismos, lenguaje coloquial... que hay que conocer para poder realizar una traducción correcta de dichas expresiones y que además pueden no tener un equivalente directo en el otro idioma.

1.3. Paradigmas de la traducción automática

Los paradigmas comunmente utilizados en traducción automática se pueden encuadrar en los niveles mostrados en la figura 1.1 [Kni97]. En la base del triángulo se muestra el paradigma de traducción más sencillo: la traducción basada en palabras. Como hemos visto antes, las palabras pueden ser ambiguas, por lo que seleccionar la traducción correcta de una determinada palabra no tiene por qué ser una tarea fácil. Además, el orden de las palabras normalmente no será el mismo en ambos idiomas. Los traductores automáticos que hagan uso de este paradigma, tienen que implementar mecanismos de algún tipo para, en cierta forma, adivinar la traducción correcta de ciertas palabras y para reordenar adecuadamente las palabras de salida. Como ejemplo de esta última capacidad, un traductor entre inglés y castellano, debería ser capaz de intercambiar el orden de adjetivos y nombres al realizar la traducción (sin conocer los conceptos de *adjetivos* ni *nombres*). A grandes rasgos, un sistema de traducción automática basado en este paradigma traduciría la frase «*the black cat*», en las siguientes fases. En primer lugar, separaría las palabras que la forman {*the*, *black*, *cat*} y generaría un conjunto de palabras como traducción de éstas, por ejemplo, {*el*, *negro*, *gato* }. A continuación, una vez tiene el conjunto de palabras que forman

Figura 1.2: Ejemplo de traducción basada en la sintaxis.

la posible traducción, las debe ordenar de alguna forma y generar una frase de salida. Una de las posibles ordenaciones daría lugar a la traducción «*el gato negro*». En la sección 2.4 se pueden ver dos modelos generativos estadísticos para la obtención de forma automática de una traducción basada en las palabras. La sección 2.5, muestra el modelo de transductores subsecuenciales, que es un sistema de traducción automática basado en este paradigma.

El paradigma de traducción automática situado en el segundo nivel de la figura 1.1 corresponde a la traducción basada en la sintaxis. La idea de éste es la siguiente: si fuéramos capaces de analizar sintácticamente la frase que queremos traducir y supiéramos cómo transferir dicha estructura sintáctica al lenguaje destino, podríamos ordenar las palabras de una forma más *inteligente*. Otra ventaja de la traducción basada en la sintaxis es que la función sintáctica de una palabra puede servir de ayuda para proponer una traducción correcta de ésta. Por ejemplo, si una misma palabra puede actuar como verbo o como sustantivo en el idioma origen y su traducción es distinta dependiendo de cuál sea su función, realizando un análisis sintáctico de la frase de entrada, podemos conocer la función sintáctica de dicha palabra y, por lo tanto, traducirla de forma consecuente. La figura 1.2 muestra el proceso de traducción de la frase «*the black cat*» utilizando el paradigma de traducción basado en la sintaxis. En primer lugar se analizaría sintácticamente la frase origen y se generaría su estructura sintáctica. A continuación, se *transferiría* dicha estructura sintáctica a la correspondiente estructura sintáctica del idioma destino. En dicha transferencia, el adjetivo y el nombre que en la frase inglesa aparecían en ese orden, en castellano invierten su posición. Por último la traducción de cada uno de los componentes de la estructura sintáctica origen se coloca en la estructura sintáctica destino y se obtiene la traducción «*el gato negro*».

Sin embargo, la traducción basada en la sintaxis genera estructuras sintácticas en el lenguaje destino que están influenciadas por la estructura sintáctica del idioma origen. Un nivel de mayor abstracción sería el mostrado en el tercer nivel de la figura 1.1. Este paradigma, el de traducción basada en la semántica, se basa en la extracción del significado de los componentes de la frase origen y la expresión de éstos en el idioma destino. Así, dada la frase «*the black cat*»

se realizaría en primer lugar un análisis semántico de ésta. El resultado de este análisis podría ser, por ejemplo, el siguiente: estamos hablando de un ser vivo, dentro de los seres vivos, de un gato y este ser vivo tiene como atributo de color de pelo, el color negro. Dado este conocimiento del significado de la frase original, se crearía la traducción en la forma en la que dicho concepto se expresaría habitualmente en el lenguaje destino.

El último nivel, correspondería a la construcción de una representación universal capaz de abarcar cualquier posible significado, llamada *Interlingua*. La traducción basada en la semántica traduce los conceptos de un idioma a otro, pero aún así, la frase destino tendría un cierto parecido con la forma en la que se expresan los conceptos en el idioma origen. Este hecho se pone más de relieve cuando los dos idiomas son muy distantes entre sí. Por ejemplo, si tradujéramos del japonés al castellano utilizando la aproximación semántica podríamos acabar con una frase del estilo: «Existe un plan de que un bebé esté ocurriendo en ella» [Kni97] que podría considerarse una buena traducción de la frase original, pero que desde luego no es la forma en la que habitualmente se diría en castellano: «Ella está embarazada». Si dispusiéramos de una Interlingua, podríamos expresar el significado completo de la frase original en un nivel mas alto de abstracción. A partir de esta versión en Interlingua, se generaría el mensaje en el idioma destino en la forma en la que habitualmente éste se expresaría en el lenguaje destino.

Una ventaja adicional, y muy atractiva, que presenta el uso de una Interlingua, es que sólo sería necesario saber cómo expresar en Interlingua los mensajes de varios idiomas y viceversa para poder traducir directamente entre cualesquiera de ellos. Cada vez que se añadiera un nuevo lenguaje, es decir, que se desarrollaran los mecanismos necesarios para expresar en Interlingua mensajes del nuevo idioma y los correspondientes para expresar en el nuevo idioma los mensajes de Interlingua, automáticamente se podría traducir desde ese lenguaje a todos los que ya estuvieran soportados y viceversa. Sin embargo, la complejidad de la definición de una Interlingua y de los procesos de conversión a y desde dicha Interlingua a un idioma dado hace esta solución sumamente compleja y difícilmente abordable.

Conforme se sube por la estructura jerárquica mostrada en la figura 1.1 se encuentran mejores ideas, generalizaciones lingüísticas y una mayor capacidad de explicación. Sin embargo, también se hace más difícil construir sistemas de traducción genéricos. Esto es debido a que los conocimientos requeridos para implementar cada uno de los paradigmas superiores se hacen demasiado grandes. En el nivel más bajo, tan sólo es necesario saber traducir determinadas palabras o expresiones de un idioma a otro. En el nivel sintáctico, para poder analizar gramaticalmente las frases que se quieren traducir, es necesario ser capaz de identificar los constituyentes de las frases y sus funciones gramaticales. En el nivel semántico, es necesario conocer el significado de todas las palabras, incluso de las más raras, y saber cómo combinar los significados de las palabras para obtener el significado de las frases. En el último nivel, es necesario disponer de un

idioma *Interlingua* en el que se pueda expresar cualquier significado del idioma origen y ser capaces de recuperar dicho significado en el idioma destino. Cuanto más arriba en la pirámide, mayor y más complejo es el conocimiento necesario para poder implementar un sistema de traducción automática.

Este conocimiento puede obtenerse mediante la deducción de una serie de reglas o por medio de ejemplos. Lo que da lugar a las dos estrategias generalmente usadas en el diseño de sistemas de traducción automática: la deductiva (basada en el conocimiento) y la inductiva (basada en ejemplos).

La *aproximación deductiva*, más tradicional, consiste en dotar al sistema de traducción de una serie de reglas (fundadas en el conocimiento de traductores expertos y que han de transferirse manualmente al sistema) que, al ser aplicadas, le permitan analizar los textos originales y, a partir de cierto punto del análisis, generar los correspondientes textos traducidos. Los sistemas así construidos suelen recibir el nombre de *sistemas de traducción basados en reglas* (STBRs). Si resultara sencillo expresar el conocimiento que poseen los traductores profesionales en forma de reglas precisas (de eliminación de ambigüedades, de transferencia, sintácticas, semánticas...) y éstas pudieran aplicarse, a la hora de traducir un texto dado, de forma computacionalmente eficiente, el problema de la traducción automática se podría considerar resuelto mediante STBRs. Sin embargo, no es éste el caso: gran parte del conocimiento de los expertos en traducción es intuitivo, de difícil o imposible formalización; los idiomas están plagados de excepciones que, como tales, no responden a reglas generales; la aplicación de ciertas reglas complejas puede resultar computacionalmente inabordable... Lo anterior serviría para explicar por qué, por ejemplo, todavía no existen buenos sistemas de traducción automática de castellano a inglés. De hecho, las traducciones obtenidas por sistemas de traducción de propósito general de una lengua a otra, sin restricciones, suelen ser de no muy buena calidad y, en la práctica, se requieren considerables dosis de intervención humana (simplificando las frases de entrada o corrigiendo las traducciones que el sistema conjetura como salida) para obtener traducciones aceptables. Por ello, más que de sistemas de traducción automática, se habla en ocasiones de sistemas de ayuda a la traducción.

A menudo, las prestaciones de los STBRs pueden mejorarse si se relaja el planteamiento generalista inicial y se centra la atención en aplicaciones específicas con dominios de discurso limitados. En este caso, no obstante, se requiere la intervención de especialistas que estudien cada nueva aplicación, bien para modelarla adecuadamente, bien para adaptar a ella un sistema de traducción de propósito general. En ambos casos suele incurrirse en costes muy elevados, por lo que sólo unos pocos de estos sistemas han llegado a ponerse en funcionamiento. Por otra parte, disponer de ellos ha servido para demostrar la utilidad práctica de los sistemas de traducción automática para aplicaciones con dominios limitados como lo es, por ejemplo, la traducción de partes metereológicos.

Por contra, los sistemas construidos siguiendo la estrategia inductiva se denominan *sistemas de traducción basados en ejemplos* (STBEs). En este caso, al sistema se le debe haber dotado de mecanismos de aprendizaje automático que

le permitan un funcionamiento en dos etapas:

1. **Aprendizaje**. A partir de un conjunto de ejemplos, que en el caso de la traducción basada en palabras estaría formada simplemente por textos en el idioma origen y su traducción en el idioma destino, el sistema elabora una representación interna del conocimiento necesario para posteriormente poder llevar a cabo traducciones similares.

2. **Traducción**. A partir del conocimiento obtenido en la fase de aprendizaje, y dado un texto en el lenguaje origen, el sistema conjetura una posible traducción de éste.

La forma más sencilla de STBE es lo que se conoce como *memoria de traducción*. El aprendizaje consiste en almacenar los ejemplos como pares de frases (original y traducción). Para realizar la traducción de una nueva frase, el sistema busca entre los ejemplos la frase origen más parecida a ésta y devuelve como respuesta su traducción. Por supuesto, existen memorias de traducción más sofisticadas que tienen en cuenta las diferencias entre la nueva frase y su ejemplo más próximo para intentar adaptar convenientemente la traducción del ejemplo encontrado. Las memorias de traducción son ampliamente utilizadas como ayuda a los traductores y es el propio traductor el que va entrenando la memoria con sus propias correcciones a las propuestas del sistema. De esta forma, la memoria sabe cómo traducir cada vez más frases y, además, se adapta al estilo de traducción del usuario. Existen variantes más complejas en las que un mismo sistema atiende las peticiones de varios traductores. En estos casos, primero busca un frase cercana a la que se quiere traducir entre los ejemplos generados por el traductor que lo está utilizando. Si no encuentra un ejemplo satisfactorio, recurre a los ejemplos proporcionados por el resto de traductores. La traducción aceptada tal cual o corregida por el traductor pasa a formar parte de su conjunto de ejemplos.

Otros sistemas de traducción automática basados en ejemplos, en lugar de memorizar los ejemplos presentados durante la fase de entrenamiento, los utilizan para la construcción de modelos de la tarea de traducción. Estos modelos pueden ser de naturaleza estadística [BDDM93], estructural [CGV94] o conexionista [CC97].

Aunque un STBE de propósito general es, hoy por hoy, impensable, la aproximación inductiva se presenta como una buena alternativa a la deductiva, en al menos dos escenarios: para auxiliar a STBRs, en módulos que han de tomar decisiones especialmente difíciles de ser formuladas mediante la aplicación de reglas precisas; y como aproximación básica en STBEs de dominio limitado. Ha de tenerse en cuenta que, una vez se dispone de un STBE, su adaptación a un nuevo par de lenguas o a un nuevo dominio es prácticamente automática. Basta con disponer de un conjunto suficientemente representativo de ejemplos de la nueva tarea de traducción.

Actualmente, gran parte de la investigación que se lleva a cabo sobre traducción automática se centra en el estudio y desarrollo de técnicas basadas en ejemplos. Estas técnicas han constituido la aproximación dominante en diferentes áreas del reconocimiento de formas y, particularmente, en el reconocimiento automático del habla. Recientemente han comenzado a popularizarse en traducción automática y en otros campos relacionados con el tratamiento del lenguaje natural.

De entre las varias aproximaciones a la traducción automática basada en ejemplos utilizando el paradigma de traducción basada en palabras, el modelo de transductores subsecuenciales [Ber79] ha sido utilizado satisfactoriamente para tareas de traducción de pequeño a medio tamaño [VJA+96, ABC+97b, CNO+].

1.4. Objetivos

La gran cantidad de muestras necesarias para producir modelos aceptables constituye el mayor problema de los sistemas de traducción automática basados en ejemplos en general y del modelo de transductores subsecuenciales en particular. Sin embargo, tal y como se muestra en [ABC+97a], un pequeño conjunto de categorías bien escogidas puede reducir substancialmente el número de ejemplos necesarios para la obtención de buenos traductores utilizando el modelo de transductores subsecuenciales. Como se ha comentado anteriormente, sería deseable poder generar también de forma automática esta categorización. En caso contrario, la reducción de costes de adaptación que es una de las ventajas del empleo de un sistema basado en ejemplos, se vería limitada por la necesidad de obtener de forma manual las categorías adecuadas a cada tarea nueva de traducción. Cuando la categorización se realiza de forma automática, y por lo tanto sin categorías definidas de antemano, recibe el nombre de agrupamiento.

El objetivo de esta tesis es el de proponer métodos de agrupamiento bilingües, de forma que los agrupamientos obtenidos por dichos métodos puedan utilizarse para la mejora de traductores automáticos basados en ejemplos. Es decir, (i) los algoritmos bilingües propuestos deberán ser capaces de agrupar palabras de forma automática en dos idiomas, (ii) el agrupamiento deberá hacerse de forma que sea útil desde un punto de vista de la traducción y (iii) este conocimiento deberá integrarse en la fase de aprendizaje de un sistema de traducción basado en ejemplos, de tal forma que se mejore la calidad de las traducciones obtenidas.

Para comprobar la adecuación al problema del método de agrupamiento bilingüe propuesto se ha elegido como sistema de traducción el modelo de transductores subsecuenciales. De todas formas, no se descarta la adaptación en el futuro de dicho método a otros sistemas de traducción, en especial a otros tipos de traductores de estados finitos.

Por tanto y debido al método que finalmente se propone de agrupamiento bilingüe, también han sido objetivos de esta tesis:

1. Proponer métodos de agrupamiento monolingüe que serán empleados por

el método de agrupamiento bilingüe propuesto.

2. Proponer un algoritmo de identificación automática de *colocaciones* (secuencias de palabras que desde el punto de vista de la traducción se comportan como una unidad) que serán utilizadas para obtener mejores agrupamientos bilingües.

Capítulo 2

Conceptos básicos

2.1. Introducción

En este capítulo presentamos la notación básica que hemos utilizado a lo largo de la presente tesis y una serie de conceptos básicos necesarios para el resto de capítulos. En particular, haremos una breve descripción de los modelos de lenguaje, del alineamiento entre palabras, de los modelos de traducción usados para la obtención automática de estos alineamientos, del modelo de transductores subsecuenciales (SSTs) y su utilización para la traducción automática y por último, de la integración de categorías en los SSTs.

Hemos pospuesto para los capítulos de agrupamiento monolingüe, agrupamiento bilingüe y colocaciones la discusión sobre los conceptos básicos referidos a cada uno de dichos temas.

El capítulo está organizado como sigue. En la primera sección introducimos la notación común al resto de capítulos y las convenciones utilizadas. El resto de notación, más específica de algún capítulo o sección en concreto, será introducida conforme sea requerida.

La sección 2.3 describe la noción de modelos de lenguaje. Explica la forma en la que pueden generarse y muestra la necesidad de su *suavizado*. También describe el método de suavizado de modelos de lenguaje utilizado en esta tesis. Estos conceptos son utilizados en los capítulos 3 de agrupamiento monolingüe y 4 de agrupamiento bilingüe. También son necesarios, aunque en menor medida, en el capítulo 5 de colocaciones.

La sección 2.4 describe el concepto de alineamiento entre palabras. Este alineamiento será utilizado en los capítulos 4 y 5 para el agrupamiento bilingüe automático y la detección de colocaciones, respectivamente. En esta sección, se

presentan también los modelos de traducción usados para la obtención automática de estos alineamientos. En concreto, se introduce el modelo generativo en el que se basan los distintos modelos de traducción presentados y se describe a grandes rasgos los métodos necesarios para la estimación de los parámetros de dichos modelos.

La sección 2.5 introduce el modelo de transductores subsecuenciales que utilizaremos para la generación de traductores automáticos. Esta tesis propone la mejora de dichos traductores por medio del agrupamiento bilingüe (capítulo 4) y la detección de colocaciones (capítulo 5).

Por último, la sección 2.6 muestra la forma en la que se pueden integrar categorías bilingües definidas de forma manual en un SST. La integración de las clases bilingües determinadas de forma automática propuesta en el capítulo 4 se basará en el método expuesto en dicha sección.

2.2. Notación básica y convenciones utilizadas

Los algoritmos propuestos en esta tesis, como se vio en el capítulo anterior, se basan en el aprendizaje a partir de ejemplos. De esta forma, podrán ser utilizados en otros idiomas con sólo disponer de ejemplos adecuados en los nuevos idiomas.

Los ejemplos necesarios son pares de frases en dos idiomas. Distinguiremos entre el idioma origen y el idioma destino. El idioma origen será aquel que queremos traducir. Y el idioma destino será aquel en el que queremos obtener la traducción. Puesto que sólo disponen de frases, la única información con la que contarán estos algoritmos es la de las palabras que forman dichas frases y la relación entre frases en un idioma y el otro. Es decir, los algoritmos no reciben información acerca de los monemas que constituyen en un determinado idioma cada palabra ni del significado de éstas. Por ejemplo, si en una frase aparece la palabra *quiero*, ésta carecerá para los algoritmos de todo significado, ya que no se habrá proporcionado información sobre que *quiero* es la primera persona del singular del verbo *querer*, ni se sabrá como descomponer cada palabra en su lexema y morfemas. En cuanto a los signos de puntuación, éstos recibirán el tratamiento de palabras.

Las frases proporcionadas como ejemplos para el aprendizaje constituyen desde el punto de vista de los algoritmos un todo en sí mismas. No es necesario que guarden relación con las frases vistas anterior o posteriormente. Los algoritmos no harán uso de conceptos de organización superiores al de la frase como podrían ser: párrafos, secciones, etc. Es decir, el contexto de una palabra vendrá dado únicamente por las palabras que forman parte de la frase a la que pertenezca.

Dicho esto, utilizaremos w para referirnos a una *palabra*. Si acompañamos a una palabra w de un subíndice, éste indicará por regla general la *posición* de la palabra dentro de una *frase*. Así la palabra w_3 correspondería a la palabra que hace tres en una frase determinada. Por otro lado, si estamos interesados en distinguir si la palabra pertenece al idioma origen o al idioma destino utilizaremos

x e y en lugar de w, respectivamente.

Representaremos una frase de cualquiera de las siguientes formas: $\bar{w}$, w_1^n o $w_1 \cdots w_n$. Los dos últimos casos los utilizaremos principalmente para poner de relieve cuántas palabras forman una frase. Además, si queremos referirnos a la secuencia de palabras que va desde la que ocupa la posición i a la que ocupa la posición j dentro de una frase utilizaremos: w_i^j o $w_i \cdots w_j$. Igual que en el caso de las palabras, si queremos matizar si una frase pertenece al idioma origen o al destino, utilizaremos x e y en lugar de w, respectivamente. Así, por ejemplo, una frase en el idioma origen se representaría como $\bar{x}$, x_1^n o $x_1 \cdots x_n$.

Un conjunto de frases en un idioma constituye para nosotros un *corpus*, al que representaremos mediante la letra $\mathcal{C}$. Diremos de este corpus que es un corpus monolingüe ya que está formado por un conjunto de frases en un sólo idioma.

Llamaremos *corpus bilingüe* a un conjunto de pares de frases, una traducción de la otra, en dos idiomas distintos. Representaremos a un corpus bilingüe en la forma $\mathcal{C}_{xy}$. En algunas ocasiones, querremos referirnos únicamente al conjunto de frases correspondiente al idioma origen del corpus $\mathcal{C}_{xy}$, para ello utilizaremos $\mathcal{C}_x$. De forma similar, para referirnos a la parte del corpus $\mathcal{C}_{xy}$ correspondiente al idioma destino utilizaremos $\mathcal{C}_y$.

Además, a lo largo de la tesis utilizaremos las siguientes convenciones. Si $\mathcal{X}$ es una variable aleatoria, representaremos la probabilidad de que la variable aleatoria $\mathcal{X}$ tome el valor x, $\mathrm{P}(\mathcal{X} = x)$, en la forma $\mathrm{P}(x)$ siempre que esto no de lugar a confusión. En las ecuaciones correspondientes a la entropía, perplejidad e información mutua utilizaremos «log» en lugar de «$\log_2$» para referirnos al logaritmo en base 2. Por último, utilizaremos la convención de que $0 \log 0 = 0$ allá donde sea necesaria.

2.3. Modelos de lenguaje

Un modelo de lenguaje permite asignar una probabilidad, $\mathrm{P}(w_1 \cdots w_n)$, a cada posible frase de un determinado idioma.

Si estuviéramos interesados en la generación de un modelo de lenguaje, podríamos desarrollar un programa que tuviera un gran conocimiento del mundo que nos rodea, de los tipos de cosas de las que la gente suele conversar, de las estructuras gramaticales utilizadas para la descripción de ciertos eventos y objetos, etc. Podríamos introducir de forma manual grandes cantidades de números en varios puntos del programa, y éstos podrían ser multiplicados o sumados en otros puntos. Esta opción parece bastante compleja. O cuando menos ardua.

Otra idea más sencilla sería la siguiente. Simplemente tendríamos que grabar todas las frases dichas por alguien en el idioma del que queremos obtener su modelo de lenguaje. ¿Cómo podríamos calcular la probabilidad de una frase? Supongamos que hubiéramos conseguido una base de datos con mil millones de frases. Si la frase *¿Cómo te va?* apareciera 76.413 veces, entonces podríamos decir que P(¿Cómo te va?) $= 76.413/1.000.000.000 = 0{,}000076413$. La grabación

podría sustituirse por la extracción automática de información de Internet o de un periódico en línea para facilitar la adquisición de una gran cantidad de datos.

Sin embargo, esta aproximación presenta el siguiente problema. La probabilidad de muchas frases perfectamente aceptables será cero simplemente porque no habrán sido vistas anteriormente.

Si nos fijamos en nuestra capacidad para juzgar si una frase es correcta o no, parece que no necesitamos recurrir a una enorme base de datos de frases ya oídas. Da la sensación de que podemos dividir una frase en una serie de componentes. Si los componentes son correctos y están combinados de una forma razonable, aceptamos la frase.

Para un computador, la manera más fácil de partir una frase en componentes es en la de fraccionarla en subcadenas. Una subcadena de n palabras recibe el nombre de *n-grama*. Si la subcadena está formada por tres palabras, es decir, $n = 3$, la llamamos *trigrama*. Si $n = 2$, *bigrama*. Si $n = 1$ puede que alguien le llame *unigrama*; la gente normal le llama *palabra*.

Un computador podría asignar una determinada probabilidad a una frase dependiendo de si sus n-gramas fueran o no razonables. Si está formada por un gran número de n-gramas razonables, puede que la frase sea correcta. No necesariamente, pero sería probable que lo fuera.

Para el caso de bigramas, podríamos definir $\mathrm{P}(w_i \mid w_{i-1})$ como la probabilidad de que la palabra w_i siga a la palabra w_{i-1}. Esta probabilidad puede estimarse utilizando un corpus de entrenamiento. Basta con dividir el número de veces que aparece la secuencia $w_{i-1}w_i$ por el número de veces que hemos visto la palabra w_{i-1} en el corpus. Cada $\mathrm{P}(w_i \mid w_{i-1})$ distinto será un *parámetro* de nuestro modelo de lenguaje.

Podemos entonces utilizar estos parámetros para obtener una aproximación de la probabilidad de una frase $w_1 \cdots w_n$:

$$\mathrm{P}(w_1 \cdots w_n) = \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_1 \cdots w_{i-1}) \simeq \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_{i-1}).$$

Para que $\mathrm{P}(w_i \mid w_{i-1})$ tenga sentido para $i = 1$, se suele añadir al comienzo de cada frase una nueva palabra, w_0, que marca el comienzo de frase (por ejemplo, <BOS>). De esta forma, todas las frases comienzan por la palabra ficticia <BOS>. Además, puesto que desearemos que la suma de las probabilidades de todas las frases $\sum_{w_1 \cdots w_n} \mathrm{P}(w_1 \cdots w_n)$ sea igual a 1, es necesario añadir una nueva palabra, <EOS>, al final de cada una de las frases e incluirla en el producto de la ecuación anterior. De lo contrario, la suma de la probabilidad de todas las frases de una determinada longitud sería 1. Y la suma de la probabilidad de todas las frases sería infinita [CG98]. Utilizando este modelo de lenguaje de bigramas podemos estimar la probabilidad de la frase *Juan lee un libro* como:

$$\begin{aligned} \mathrm{P}(\text{Juan lee un libro}) \quad \simeq \quad & \mathrm{P}(\text{Juan} \mid \text{<BOS>}) \cdot \mathrm{P}(\text{lee} \mid \text{Juan}) \cdot \mathrm{P}(\text{un} \mid \text{lee}) \cdot \\ & \cdot \mathrm{P}(\text{libro} \mid \text{un}) \cdot \mathrm{P}(\text{<EOS>} \mid \text{libro}). \end{aligned}$$

O lo que es lo mismo, la probabilidad de la frase *Juan lee un libro* vendrá dada por la respuesta a las siguientes preguntas. ¿Cuál es la posibilidad de que una frase comience por la palabra *Juan*? Si la frase comenzó por *Juan*, ¿cuál es la posibilidad de que la siguiente palabra sea *lee*? Si esta palabra fue *lee*, ¿es razonable que la palabra *un* sea la siguiente palabra? Y así sucesivamente.

En principio, parece interesante que los n-gramas sean cuanto más grandes mejor. De esta forma, se podrían modelar dependencias entre palabras de una misma frase muy distantes entre sí. Sin embargo, cuanto mayor es n, mucho mayor es el número de parámetros que deben ser estimados. Por ejemplo, si el vocabulario del lenguaje que queremos modelar fuera de tan sólo 500 palabras (una tarea de dominio restringido), utilizando bigramas tendríamos que estimar $500 \times 500 = 250.000$ parámetros. Para el caso de trigramas, $500^3 = 12{,}5$ millones. Si nos vamos a cuatro-gramas la cifra sería de $500^4 = 62.500$ millones.

Es decir, cuanto más grande sea el n-grama utilizado, menos práctico resulta. Aunque dispusiéramos de un corpus enorme, el número de parámetros que se deben estimar superará con creces al número de ejemplos disponibles. Por esta razón, normalmente se utilizan modelos de bigramas o trigramas, que, en la práctica, y en contra de lo que podría parecer desde un punto de vista lingüístico, resultan ser suficientemente buenos predictores.

Para la obtención de los parámetros $\mathrm{P}(v \mid u)$ hemos utilizado anteriormente lo que se conoce como estimación por máxima-verosimilitud. Es decir, $\mathrm{P}(v \mid u)$ es estimado como $\mathrm{n}(uv)/\mathrm{n}(u)$, donde $\mathrm{n}(uv)$ es el número de veces que en el corpus de entrenamiento se ha visto la secuencia de palabras uv y $\mathrm{n}(v)$ es el número de veces que se ha visto en el corpus de entrenamiento la palabra v. Esta estimación es el resultado de asignar los valores de los distintos parámetros de tal forma que la verosimilitud, V, de la muestra empleada para su entrenamiento sea máxima. La verosimilitud, V, se define como la probabilidad conjunta de la muestra, que utilizando el modelo de bigramas se expresa como:

$$V = \mathrm{P}(w_1 \cdots w_n) = \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_{i-1}). \tag{2.1}$$

Puesto que maximizar V equivale a maximizar $\log(V)$, definiremos V_l como:

$$\begin{aligned} V_l &= \log\left(\prod_{i=1}^{n} \mathrm{P}(w_i \mid w_{i-1})\right) \\ &= \sum_{i=1}^{n} \log(\mathrm{P}(w_i \mid w_{i-1})) \\ &= \sum_{uv} \mathrm{n}(uv) \log(\mathrm{P}(v \mid u)), \end{aligned} \tag{2.2}$$

donde $\mathrm{n}(uv)$, como se ha visto antes, es el número de veces que se ha visto en la muestra la secuencia de palabras uv.

Al seleccionar los distintos $\mathrm{P}(v \mid u)$ de tal forma que se optimice V_l tenemos que respetar la siguiente restricción. La suma de las probabilidades condicionales para cualquier u tiene que ser igual a uno:

$$\sum_v \mathrm{P}(v \mid u) = 1,\ \forall u. \tag{2.3}$$

Por lo tanto, estamos ante un problema de optimización de una función, expresada en (2.2), sujeta a las anteriores restricciones. Para resolverlo utilizaremos, como es habitual en estos casos, el método de los multiplicadores de Lagrange [YB97, págs. 179–180]. Para ello, en primer lugar consideraremos la función $\tilde{V}_l$:

$$\tilde{V}_l = \sum_{uv} \mathrm{n}(uv) \log(\mathrm{P}(v \mid u)) - \sum_u \lambda_u \left[\sum_v \mathrm{P}(v \mid u) - 1 \right], \tag{2.4}$$

donde λ_u son los multiplicadores de Lagrange. Estos son simplemente constantes, de los que desconocemos su valor, y que vamos a utilizar para la obtención del resto de parámetros.

A continuación, siguiendo el método de los multiplicadores de Lagrange, tomaremos las derivadas parciales de $\tilde{V}_l$ con respecto a cada $\mathrm{P}(v \mid u)$ y a cada multiplicador de Lagrange, λ_u, y las igualaremos a 0. Con lo que obtendremos un conjunto de ecuaciones de los siguientes tipos:

$$\begin{aligned} \frac{\partial \tilde{V}_l}{\partial \mathrm{P}(v \mid u)} &= \frac{\mathrm{n}(uv)}{\mathrm{P}(v \mid u)} - \lambda_u = 0, \\ \frac{\partial \tilde{V}_l}{\partial \lambda_u} &= \sum_v \mathrm{P}(v \mid u) - 1 = 0. \end{aligned}$$

Conviene notar que, la segunda ecuación representa simplemente la restricción de normalización de la probabilidad para cada palabra u ya expresada en (2.3). Mediante una serie de sencillas manipulaciones sobre las ecuaciones anteriores llegamos a:

$$\mathrm{P}(v \mid u) = \frac{\mathrm{n}(uv)}{\sum_{v'} \mathrm{n}(uv')} = \frac{\mathrm{n}(uv)}{\mathrm{n}(u)} \tag{2.5}$$

Por lo tanto, la estimación por máxima-verosimilitud de $\mathrm{P}(v \mid u)$ resulta ser la frecuencia relativa $\mathrm{n}(uv)/\mathrm{n}(u)$, tal y como queríamos demostrar. Normalmente, se suele justificar la adopción de dichas frecuencias relativas mediante una interpretación intuitiva del concepto de probabilidad. Sin recurrir al marco matemático mostrado que lo justifica.

Esta estimación de $\mathrm{P}(v \mid u)$ por máxima-verosimilitud presenta sin embargo el siguiente problema: asigna una probabilidad cero a los eventos no vistos (tal y como se ha visto que ocurría en el ejemplo de modelo de lenguaje basado en frases

completas). Puesto que la probabilidad de una frase se calcula como el producto de las probabilidades de sus n-gramas, estos ceros se propagarán y darán una mala estimación de la probabilidad (probabilidad cero) de aquellas frases que contengan n-gramas no vistos previamente en el corpus de entrenamiento. Por ejemplo, si en nuestro corpus de entrenamiento no hubiéramos visto el bigrama *Pepe lee*, la probabilidad de P(lee | Pepe) estimada por máxima-verosimilitud sería cero. Por lo tanto, la estimación de la probabilidad de la frase *Pepe lee un libro*, por lo demás correcta, sería también cero. Lo que a todas luces no parece una buena estimación.

Para evitar esto, es necesario utilizar mejores estimadores. Estimadores que tengan en cuenta que aquellos eventos que no han sido vistos en el corpus de entrenamiento podrían darse en la práctica. La mayoría de estos estimadores se basan en la reducción de una u otra forma de la probabilidad asignada a los eventos vistos previamente. Así, esta masa de probabilidad *descontada* de los eventos vistos se puede repartir entre los eventos no vistos.

Los métodos por los que se obtienen estos estimadores suelen recibir el nombre de métodos de descuento (*discounting methods*) y el proceso mediante el cual se realiza el descuento suele denominarse suavizado (*smoothing*).

Conviene tener presente que la necesidad de utilizar modelos de lenguaje suavizados es el resultado de los siguientes condicionantes [YB97, pág. 177]:

- Cualquier combinación de palabras debe ser posible, esto es, no tiene que haber ninguna secuencia de palabras que tenga una probabilidad exactamente igual a 0.
- La cantidad de datos de entrenamiento disponible para la estimación de dichos modelos es siempre insuficiente, incluso si el corpus de entrenamiento consta de varios cientos de millones de palabras.
- Considerar a los eventos no vistos (en los datos de entrenamiento) cuando se utiliza validación cruzada (ver la sección 3.5).

Haremos uso de modelos de lenguaje en los métodos para agrupamiento de palabras descritos en las secciones 3.4 y 3.5. En realidad, utilizaremos modelos de lenguaje basados en clases. Y sólo para la obtención del agrupamiento. Es decir, durante la fase de entrenamiento. Sin pretender entrar en detalles ahora, el algoritmo propuesto en la sección 3.4, utiliza un número de clases fijado de antemano y asegura que al menos haya una palabra asignada a cada clase. Esto implica que no pueden darse eventos no vistos (durante la fase de entrenamiento que es donde se utiliza el modelo) y por lo tanto no será necesario suavizar el modelo de lenguaje basado en clases que se utiliza.

Sin embargo, en el algoritmo propuesto en la sección 3.5 se utiliza una extensión del método de validación cruzada que sí que hace necesario el uso de un método de suavizado.

En particular, el método de suavizado que hemos utilizado es el método de descuento absoluto propuesto por Ney y Essen [NE93]. El concepto clave

que se maneja en dicho método es el de cuenta, r, o el número de veces que un determinado evento aparece en el corpus. La idea principal del método es que una vez suavizadas todas las cuentas, las más altas queden prácticamente iguales. La justificación para este método es la siguiente. El número de veces, $r = \mathrm{n}(uv)$, que una determinada secuencia ha sido vista en un conjunto de datos de entrenamiento es posible que cambie si contamos la aparición de dicha secuencia en otro conjunto de datos del mismo tamaño. Sin embargo, es de esperar que la diferencia entra las dos observaciones sea pequeña. Típicamente esperaríamos ver valores como $r-1$, r, $r+1$. Para tener en cuenta esta variación, Ney y Essen introducen un valor medio, b, para la reducción de todas las cuentas, que se asume independiente del valor de la cuenta r. Al descontar el mismo valor b, donde $0 < b < 1$, a todas las cuentas, las cuentas más altas quedaran prácticamente inalteradas.

El método consistirá por lo tanto en descontar de todas las cuentas distintas de cero una pequeña cantidad constante b, y repartir de forma uniforme la frecuencia así ganada entre los eventos no vistos. Si se aplica este método para suavizar la estimación de la probabilidad conjunta[1] (que es para lo que se utiliza en esta tesis), entonces:

$$\mathrm{P}(uv) = \begin{cases} \dfrac{\mathrm{n}(uv) - b}{N} & \text{si } \mathrm{n}(uv) > 0 \\ \dfrac{\mathrm{n}_+ b}{\mathrm{n}_0 N} & \text{si } \mathrm{n}(uv) = 0 \end{cases} \tag{2.6}$$

donde N es el número de bigramas en el corpus de entrenamiento, n_+ el número de bigramas que se ha visto al menos una vez y n_0 el número de posibles bigramas que no han sido vistos en el corpus de entrenamiento. Si nos fijamos en la anterior ecuación, $\mathrm{n}_+ b/N$ es la masa de probabilidad descontada a los n_+ eventos vistos. Esta masa de probabilidad se reparte entre el número de eventos no vistos n_0 de forma uniforme. Por lo tanto, la probabilidad de un evento no visto se estima como $\mathrm{n}_+ b/n_0 N$.

Otros métodos de suavizado comunmente utilizados pueden verse en [MS00].

2.4. Alineamiento y modelos de traducción

Los algoritmos propuestos en esta tesis de agrupamiento bilingüe (capítulo 4) y de colocaciones (capítulo 5), hacen uso del concepto de *alineamiento*.

Dado un par de frases $(\bar{x}, \bar{y})$, una traducción de la otra, podemos considerar que cada palabra y pueda ser debida a la traducción de alguna palabra x. Definiremos el alineamiento, $\vec{a}$, entre un par de frases, $(\bar{x}, \bar{y})$, como un vector que indica para cada palabra y aquella palabra x de la que ha *surgido*. La figura 2.1 muestra una posible representación de un alineamiento de este tipo. En

[1] El método de descuento absoluto aplicado a la estimación de la probabilidad condicional se puede consultar en [YB97, pág. 188].

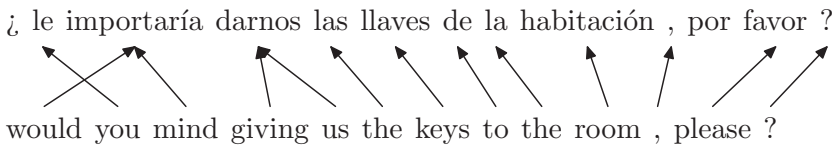


Figura 2.1: Ejemplo de alineamiento entre un par de frases.

este alineamiento se puede ver que las palabras *would* y *mind* han sido alineadas con *importaría*, la palabra *you* con *le*, y así sucesivamente. También podemos ver que es posible que haya palabras x con las que no esté alineada ninguna palabra y. En el ejemplo, las palabras «*¿*» y *por*. De igual forma, se permite, aunque no haya ningún caso en el ejemplo, que haya palabras y que no estén alineadas con ninguna palabra x. En tal caso, diremos que están alineadas con la palabra vacía, a la que denotaremos como λ.

Esta definición de alineamiento presenta algunas deficiencias con respecto a las posibles relaciones existentes entra las palabras de dos frases. Por ejemplo, la palabra inglesa *please* aparece alineada con *favor* pero sería perfectamente aceptable que hubiera estado alineada con la palabra *por*. De hecho, *please* debería estar alineada simultáneamente con *por* y con *favor*. El alineamiento propuesto no permite que una palabra y sea alineada simultáneamente con dos palabras x. Más genéricamente, podría ocurrir que varias palabras de la frase de salida estuvieran alineadas como un todo con varias palabras de la frase de entrada que también actuaría como un todo. Podríamos considerar que un alineamiento más *razonable* haría que la expresión «*would you mind*» estuviera alineada con la expresión «*le gustaría*», en lugar de considerar los alineamientos a nivel de palabras. Pese a estas limitaciones, nos restringiremos en esta tesis a esta definición de alineamiento.

Para representar el alineamiento de una forma más compacta utilizaremos un vector de alineamientos, $\vec{a}$. En este vector indicaremos para cada palabra j de la frase de salida $\bar{y}$ la posición de la palabra x en la frase $\bar{x}$ con la que está alineada. Así, el alineamiento de la figura 2.1 lo podemos representar como $\vec{a} = [3, 2, 3, 4, 4, 5, 6, 7, 8, 9, 10, 12, 13]$. De esta forma, el elemento a_j indicará la posición con la que está alineada la palabra y_j. Así, a_1 indica la posición de la palabra x con la que está alineada y_1=*would*, es decir, $x_{a_j} = x_3$=*importaría*. También es práctica común representar la traducción junto con el alineamiento de la siguiente forma: *(¿ le importaría darnos las llaves de la habitación , por favor ? | would (3) you (2) mind (3) giving (4) us (4) the (5) keys (6) to (7) the (8) room (9) , (10) please (12) ? (13))*. Por convención, se suele asignar a la palabra vacía, λ, la posición 0 en la frase de entrada. En consecuencia, representaremos que una palabra y_i está alineada con la palabra vacía haciendo que $a_i = 0$.

Brown et al. sientan en [BDDM93] las bases de una serie de modelos matemáticos para la estimación de modelos de traducción estadísticos. Utilizando

dichos modelos se puede obtener de forma automática el alineamiento $\vec{a}$ para cada par de frases $(\bar{x}, \bar{y})$. En esta tesis hemos utilizado estos modelos principalmente para el alineamiento automático del corpus de entrenamiento. En particular, este alineamiento será utilizado por los algoritmos descritos en los capítulos 4 y 5 para la generación de agrupamientos bilingües y de colocaciones, respectivamente. Algunos alineamientos automáticos obtenidos para la frase de la figura 2.1 utilizando varios de estos modelos se muestran en la sección 4.3.1.

En el contexto de la traducción estadística, cada frase de un determinado idioma, $\bar{y}$, se considera como una posible traducción de una frase en otro idioma, $\bar{x}$. Cada par de frases $(\bar{x}, \bar{y})$ tiene asignado un número, $\mathrm{P}(\bar{y} \mid \bar{x})$, que se interpreta como la probabilidad de que un traductor, si se le presenta $\bar{x}$, genere $\bar{y}$ como su traducción. Llamaremos a este modelo de traducción *modelo de traducción directo*.

De igual forma, podemos plantearnos el problema a la inversa: si sabemos que un traductor ha generado la frase $\bar{y}$ como traducción de alguna frase desconocida $\bar{x}$ con una probabilidad $\mathrm{P}(\bar{y} \mid \bar{x})$, ¿podemos calcular de alguna forma la probabilidad de que una determinada $\bar{x}$ sea la que ha sido traducida como $\bar{y}$? Llamaremos a este modelo de traducción *modelo de traducción inverso*. El propósito del método propuesto en Brown et al. será descubrir aquella $\bar{x}$ que con más probabilidad ha generado la frase $\bar{y}$. Es decir, escogen aquella frase $\bar{x}'$ para la que la $\mathrm{P}(\bar{x} \mid \bar{y})$ es máxima:

$$\mathrm{P}(\bar{x} \mid \bar{y}) = \frac{\mathrm{P}(\bar{x})\mathrm{P}(\bar{y} \mid \bar{x})}{\mathrm{P}(\bar{y})}. \tag{2.7}$$

Puesto que el denominador es independiente de $\bar{x}$, encontrar $\bar{x}'$ equivale a encontrar aquella frase $\bar{x}$ de tal forma que el producto $\mathrm{P}(\bar{x})\mathrm{P}(\bar{y} \mid \bar{x})$ sea tan grande como sea posible. Se llega entonces a la que es la ecuación fundamental de la traducción estadística:

$$\bar{x}' = \arg\max_{\bar{x}} \mathrm{P}(\bar{x})\mathrm{P}(\bar{y} \mid \bar{x}). \tag{2.8}$$

Puede parecer un proceso un poco extraño. Para obtener la traducción de $\bar{y}$, es necesario calcular para todas las posibles frases $\bar{x}$, el producto de la probabilidad *a priori* de $\bar{x}$, $\mathrm{P}(\bar{x})$, y la probabilidad condicional de la frase $\bar{y}$ dada la frase $\bar{x}$, $\mathrm{P}(\bar{y} \mid \bar{x})$.

¿Por qué estimar $\mathrm{P}(\bar{x})$ y $\mathrm{P}(\bar{y} \mid \bar{x})$ cuando realmente queremos obtener $\mathrm{P}(\bar{x} \mid \bar{y})$? ¿Por qué no estimar directamente $\mathrm{P}(\bar{x} \mid \bar{y})$? Para entender esto es necesario diferenciar las posibles frases $\bar{x}$ e $\bar{y}$, entre frases que están bien o mal formadas. Este no es un concepto preciso, ciertas frases son *aceptables* en un determinado lenguaje mientras que otras no. Si son *aceptables* diremos que están bien formadas. Cuando traducimos de forma directa del idioma de $\bar{y}$ al idioma de $\bar{x}$, partimos de una frase bien formada $\bar{y}$ y esperamos acabar en una frase bien formada $\bar{x}$. Por lo tanto, será importante que el modelo directo para $\mathrm{P}(\bar{x} \mid \bar{y})$

$\bar{x}$ = quiero una habitación doble .	
$\bar{y}$ = I want a double room .	$\bar{y}'$= I want a room double .
Modelo directo: $\mathrm{P}(\bar{y} \mid \bar{x}) = 0{,}00047042$	Modelo directo: $\mathrm{P}(\bar{\mathbf{y}}' \mid \bar{\mathbf{x}}) = \mathbf{0{,}0062681}$
Modelo inverso: $\mathrm{P}(\bar{x} \mid \bar{y}) = 0{,}0018831$ $\mathrm{P}(\bar{y}) = 0{,}00053674$ $\mathrm{P}(\bar{\mathbf{y}}) \cdot \mathrm{P}(\bar{\mathbf{x}} \mid \bar{\mathbf{y}}) = \mathbf{1{,}0107 \cdot 10^{-6}}$	Modelo inverso: $\mathrm{P}(\bar{x} \mid \bar{y}') = 0{,}020263$ $\mathrm{P}(\bar{y}') = 4{,}9756 \cdot 10^{-10}$ $\mathrm{P}(\bar{y}') \cdot \mathrm{P}(\bar{x} \mid \bar{y}') = 1{,}0082 \cdot 10^{-11}$

Cuadro 2.1: Probabilidad de una traducción según los modelos directo e inverso.

concentre su probabilidad tanto como sea posible en las frases $\bar{x}$ bien formadas. Por el contrario, si utilizamos el modelo inverso, no es importante que $\mathrm{P}(\bar{y} \mid \bar{x})$ concentre su probabilidad en frases $\bar{y}$ bien formadas ya que esto no afectará a la calidad de las traducciones (queremos generar una frase $\bar{x}$ como traducción de $\bar{y}$, no al revés). Los modelos propuestos en [BDDM93] para la estimación de $\mathrm{P}(\bar{y} \mid \bar{x})$ acaban esparciendo la probabilidad por todas partes, mucha de ella sobre frases $\bar{y}$ mal formadas, por lo que no podrían utilizarse para la estimación del modelo directo $\mathrm{P}(\bar{x} \mid \bar{y})$.

Es más, cuando utilizamos el modelo inverso de traducción, los dos factores en (2.8) cooperan. La probabilidad del modelo de traducción, $\mathrm{P}(\bar{y} \mid \bar{x})$, será grande para aquellas frases $\bar{y}$, bien o mal formadas, que contengan las palabras necesarias en aproximadamente los lugares adecuados como para explicar la frase $\bar{x}$. La probabilidad del modelo de lenguaje, $\mathrm{P}(\bar{x})$, será grande para aquellas frases $\bar{x}$ bien formadas, independientemente de su relación con la frase $\bar{y}$. Juntas, producen una probabilidad grande para aquellas frases $\bar{x}$ bien formadas que explican bien $\bar{y}$. No se puede conseguir esto utilizando un modelo de traducción directo.

El cuadro 2.1 muestra los valores obtenidos utilizando modelos directos e inversos de traducción para dos traducciones de la frase $\bar{x}$=«*quiero una habitación doble .*». En la columna de la izquierda se muestran los resultados obtenidos si la traducción es $\bar{y}$=«*I want a double room .*», que es una traducción correcta. En la columna de la derecha se muestran los resultados obtenidos si la traducción considerada es $\bar{y}'$=«*I want a room double .*» en la que las palabras *room* y *double* no aparecen en el orden *correcto* para el inglés. Como se puede ver, si utilizáramos un modelo directo de traducción, $\mathrm{P}(\bar{y}' \mid \bar{x})$ es mayor que $\mathrm{P}(\bar{y} \mid \bar{x})$, por lo que propondríamos erróneamente como traducción de $\bar{x}$, la frase $\bar{y}'$.

Sin embargo, cuando utilizamos el modelo de traducción inverso, podemos observar que pese a la probabilidad de traducción para $\bar{y}'$, $\mathrm{P}(\bar{y}' \mid \bar{x})$, sigue siendo mayor que $\mathrm{P}(\bar{y} \mid \bar{x})$, la probabilidad del modelo de lenguaje para $\bar{y}$, $\mathrm{P}(\bar{y})$, es mucho mayor que la probabilidad del modelo de lenguaje para $\bar{y}'$, $\mathrm{P}(\bar{y}')$. Y debido a esto, la probabilidad del modelo de traducción inverso para $\bar{y}$, $\mathrm{P}(\bar{y}) \cdot \mathrm{P}(\bar{x} \mid \bar{y})$,

es mucho mayor que para $\bar{y}'$, $\mathrm{P}(\bar{y}') \cdot \mathrm{P}(\bar{x} \mid \bar{y}')$. A la vista de los valores de estas probabilidades, propondríamos acertadamente $\bar{y}$ como la traducción de $\bar{x}$.

Los valores mostrados en el cuadro 2.1 se han obtenido de la siguiente forma. Se ha utilizado como modelo de traducción el modelo 2 de [BDDM93]. Éste se ha entrenado mediante 10 iteraciones del modelo 1 y 10 iteraciones del modelo 2 sobre el corpus EuTrans I (ver capítulo 6). En cuanto al modelo de lenguaje, se ha utilizado un trigrama suavizado entrenado también sobre el corpus EuTrans I.

Volviendo a la estimación de los modelos de traducción, es interesante destacar que la ecuación (2.8) resume los tres retos computacionales que presenta la traducción estadística en la práctica: (i) estimar la probabilidad de un modelo de lenguaje $\mathrm{P}(\bar{x})$, (ii) estimar la probabilidad de un modelo de traducción $\mathrm{P}(\bar{y} \mid \bar{x})$ y (iii) concebir una búsqueda eficiente capaz de encontrar la frase $\bar{x}'$ que maximice su producto.

En [BDDM93], Brown et al., se centran en el problema de modelar la probabilidad de traducción $\mathrm{P}(\bar{y} \mid \bar{x})$. Para ello presentan cinco modelos, a los que llaman en orden creciente de complejidad modelo 1 al modelo 5. Nos referiremos a ellos como los modelos IBMm1 al IBMm5.

Cada uno de estos modelos define un procedimiento mediante el cual calcular la probabilidad condicional $\mathrm{P}(\bar{y} \mid \bar{x})$ que definen como la verosimilitud de la traducción $(\bar{x}, \bar{y})$. Esta verosimilitud es una función que depende de un gran número de parámetros que deben estimarse durante la fase de entrenamiento. Dado un conjunto de traducciones, la verosimilitud de este conjunto se puede calcular como el producto de la verosimilitud de sus miembros.

A grandes rasgos, la forma de proceder de estos modelos consiste en adivinar en primera instancia un valor inicial para estos parámetros y entonces aplicar el algoritmo *EM* (*Expectation Maximization*) en sucesivas iteraciones buscando llegar a un máximo de la verosimilitud del conjunto de traducciones de entrenamiento. Hay que tener en cuenta que si la verosimilitud presenta varios máximos locales, el máximo al que se llegue dependerá de cuales hayan sido los valores elegidos inicialmente.

Una característica de esta serie de modelos es que el conjunto de parámetros estimado por uno de ellos puede ser utilizado para construir una estimación inicial del conjunto de parámetros del siguiente. De esta forma, cuando en esta tesis nos refiramos por ejemplo al alineamiento generado por el modelo IBMm4, en realidad dicho alineamiento se ha obtenido tras varias iteraciones de los modelos IBMm1, IBMm2, IBMm3 y por último del modelo IBMm4.

Los modelos IBMm1 e IBMm2 presentan un formulación especialmente sencilla que permite que las iteraciones del algoritmo EM sean calculadas de forma exacta. Además, el modelo IBMm1 presenta un único máximo, por lo que los parámetros obtenidos tras una serie de iteraciones del algoritmo EM no depende de la suposición inicial acerca del valor de estos parámetros. El modelo IBMm2 y los siguientes no poseen un único máximo local, pero si se inicializan los parámetros de cada modelo a partir de los parámetros del modelo anterior, se llega a un modelo que no depende de cual haya sido la inicialización de los parámetros

del modelo IBMm1.

La probabilidad $\mathrm{P}(\bar{y} \mid \bar{x})$ se puede expresar en función de la probabilidad condicional $\mathrm{P}(\bar{y}, \vec{a} \mid \bar{x})$ como:

$$\mathrm{P}(\bar{y} \mid \bar{x}) = \sum_{\vec{a}} \mathrm{P}(\bar{y}, \vec{a} \mid \bar{x}), \tag{2.9}$$

donde $\vec{a}$ es el alineamiento en la forma definida anteriormente. Los modelos IBMm1 al IBMm5 proponen distintos métodos para la estimación de $\mathrm{P}(\bar{y}, \vec{a} \mid \bar{x})$. Además, los modelos IBMm1 e IBMm2 utilizan un modelo generativo distinto al de los modelos IBMm3, IBMm4 e IBMm5.

En los modelos IBMm1 e IBMm2, la forma en la que se supone que se genera una frase $\bar{y}$ y el alineamiento $\vec{a}$ partiendo de una frase $\bar{x}$ es la siguiente. En primer lugar se escoge la longitud de la frase de salida, J, en función del conocimiento de la frase $\bar{x}$. Luego, se escoge con qué posición de la frase $\bar{x}$ se alineará la primera palabra y_1 dado el conocimiento de la frase $\bar{x}$ y de la longitud de la frase que se está generando, J. A continuación, se elige la identidad de la palabra y_1 dados el conocimiento de la frase $\bar{x}$, la longitud de la frase que se está generando, J, y con qué posición de la frase $\bar{x}$ se ha alineado esta palabra. Y así sucesivamente. Conforme se avanza en la generación de la frase $\bar{y}$, la elección en cada punto se realizará en función del conocimiento completo de la frase $\bar{x}$ y el de todas las anteriores elecciones realizadas en cuanto a la frase $\bar{y}$ y sus alineamientos. La siguiente ecuación muestra esta forma de generar $\bar{y}$ partiendo de $\bar{x}$:

$$\mathrm{P}(\bar{y}, \vec{a} \mid \bar{x}) = \mathrm{P}(J \mid \bar{x}) \prod_{j=1}^{J} \mathrm{P}(a_j \mid a_1^{j-1}, y_1^{j-1}, J, \bar{x}) \mathrm{P}(y_j \mid a_1^{j}, y_1^{j-1}, J, \bar{x}). \tag{2.10}$$

Veamos con un ejemplo cómo se generaría una posible traducción de la frase «*quiero una habitación doble .*». En primer lugar escogeríamos una longitud de la frase de salida conociendo la frase de entrada. Pongamos que decidimos que dicha longitud sea de 6 palabras (recordemos que los signos de puntuación cuentan como palabras). Nuestra traducción, por tanto, sólo tendría de momento huecos donde más tarde irán las palabras de la frase de salida:

$\bar{x}$ =*quiero una habitación doble .*

$\bar{y}$ =____, ____, ____, ____, ____, ____,

A continuación, escogeríamos con qué palabra se alinea la primera palabra de la frase $\bar{y}$. Supongamos que elegimos que se alinee con la x_1:

$\bar{y}$ =____ *(1)* ____, ____, ____, ____, ____,

Conociendo la frase $\bar{x}$, que la longitud de $\bar{y}$ es 6 y sabiendo que y_1 está alineada con x_1, elegimos una palabra para y_1. Supongamos que esta palabra sea *I*:

$\bar{y}$ =*I (1)* ____, ____, ____, ____, ____,

Si seguimos el mismo procedimiento para las restantes palabras podríamos realizar las siguientes elecciones:

$\bar{y}$ =*I (1)* ____ *(1)* ____, ____, ____, ____,

$\bar{y}$ = *I (1) want (1)* ___ ___ ___ ___
$\bar{y}$ = *I (1) want (1)* ___ *(2)* ___ ___ ___
$\bar{y}$ = *I (1) want (1) a (2)* ___ ___ ___
...
$\bar{y}$ = *I (1) want (1) a (2) double (4) room (3) . (5)*

Por lo que la traducción generada quedaría: «*I want a double room .*».

Una vez visto el modelo generativo empleado por los modelos IBMm1 e IBMm2 cabe preguntarse como estiman estos modelos las probabilidades condicionales que aparecen en la parte derecha de la ecuación (2.10). Estas probabilidades no pueden considerarse todas como parámetros independientes simplemente porque hay demasiadas. En el modelo IBMm1, se realizan las siguientes suposiciones para simplificar el cálculo de $\mathrm{P}(\bar{y}, \vec{a} \mid \bar{x})$. Se asume que $\mathrm{P}(J \mid \bar{x})$ es independiente de J y $\bar{x}$. Es decir, cualquier longitud *posible* de $\bar{y}$ tendrá la misma probabilidad. Ésta vendrá dada por un parámetro al que denominan $\epsilon \equiv \mathrm{P}(J \mid \bar{x})$. Naturalmente, la longitud de la frase de salida sí que de suponer que guarde alguna relación con la frase de entrada, al menos con su longitud. El objetivo de ésta y de las demás simplificaciones del modelo IBMm1 es llegar a una expresión matemática sencilla que genere una función de verosimilitud con un único máximo local. De esta forma, el proceso de alcanzar dicho máximo será independiente de los valores iniciales.

También se asume que $\mathrm{P}(a_j \mid a_1^{j-1}, y_1^{j-1}, J, \bar{x})$, depende sólo de I, la longitud de $\bar{x}$. Si la estimación de esta probabilidad sólo depende de I, la probabilidad de que una determinada palabra y_j esté alineada con cualquier palabra de $\bar{x}$ o con la palabra vacía, será la misma para todas ellas y vendrá dada por tanto por $(I+1)^{-1}$.

Por último, se supone que $\mathrm{P}(y_j \mid a_1^j, y_1^{j-1}, J, \bar{x})$ depende sólo de y_j y de x_{a_j}, dando lugar al parámetro $t(y_j \mid x_{a_j}) \equiv \mathrm{P}(y_j \mid a_1^j, y_1^{j-1}, J, \bar{x})$.

Utilizando las anteriores suposiciones, el modelo IBMm1 reescribe (2.10) como:

$$\mathrm{P}(\bar{y}, \vec{a} \mid \bar{x}) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} t(y_j \mid x_{a_j}). \tag{2.11}$$

Por lo que (2.9) queda:

$$\mathrm{P}(\bar{y} \mid \bar{x}) = \frac{\epsilon}{(I+1)^J} \sum_{a_1=0}^{I} \cdots \sum_{a_J=0}^{I} \prod_{j=1}^{J} t(y_j \mid x_{a_j}). \tag{2.12}$$

Se quiere obtener el valor de los parámetros $t(y \mid x)$ de tal forma que se maximice $\mathrm{P}(\bar{y} \mid \bar{x})$, respetando, además, la restricción de que para cada x se cumpla:

$$\sum_{y} t(y \mid x) = 1. \tag{2.13}$$

Puesto que se tiene una función que se quiere maximizar sujeta a restricciones se puede utilizar el método de los multiplicadores de Lagrange (de forma similar a la ya vista en la sección 2.3). Haciendo esto se llega a una serie de ecuaciones en las que $t(y \mid x)$ aparece tanto en la parte izquierda como en la derecha de éstas. Por lo tanto, no se puede resolver directamente el valor de los parámetros $t(y \mid x)$. Sin embargo, se puede utilizar el siguiente procedimiento iterativo para llegar a una solución: dada una serie de valores iniciales de $t(y \mid x)$, se puede evaluar la parte izquierda de la ecuación en función de estos valores iniciales y utilizar el resultado como nuevos estimadores de $t(y \mid x)$. Este proceso, cuando se aplica repetidamente, recibe el nombre de algoritmo EM. En el caso del modelo IBMm1, puesto que (2.12) presenta un único máximo, el algoritmo EM converge y encuentra el máximo sin importar cuales sean los valores iniciales escogidos.

En cuanto al modelo IBMm2, en éste se realizan las mismas suposiciones que en el IBMm1 salvo por el hecho de que se asume que $\mathrm{P}(a_j \mid a_1^{j-1}, y_1^{j-1}, J, \bar{x})$, además de depender de I, depende de j, a_j, y J, por lo que se introducen un conjunto de probabilidades de alineamiento:

$$a(a_j \mid j, J, I) \equiv \mathrm{P}(a_j \mid a_1^{j-1}, y_1^{j-1}, J, \bar{x}). \tag{2.14}$$

El procedimiento seguido a partir de aquí es similar al del modelo IBMm1 y será el utilizado también en los restantes modelos. A grandes rasgos, se especifica una serie de restricciones que han de cumplir los parámetros. Se aplica el método de los multiplicadores de Lagrange para resolver el máximo de la verosimilitud obtenida por el modelo en cuestión, sujeta a las restricciones impuestas. Y por último, se aplica el algoritmo EM para la estimación de los parámetros. En los modelos IBMm3 al IBMm5, a diferencia de los modelos IBMm1 e IBMm2, no se pueden tener en cuenta todos los posibles alineamientos debido a que la cantidad de operaciones que habría que realizar haría el problema intratable. Por ello, en el algoritmo EM intervendrán únicamente aquellos alineamientos que son considerados más probables.

El resto de detalles correspondiente a la implementación de los distintos modelos se puede encontrar en [BDDM93]. Simplemente mostraremos a continuación el modelo generativo utilizado por los modelos IBMm3, IBMm4 e IBMm5. Dada una frase $\bar{x}$, para generar una posible traducción $\bar{y}$, en primer lugar se decide la *fertilidad*, ϕ_i, de cada palabra x_i, $\mathrm{P}(\phi_i \mid x_i)$. La fertilidad se define como el número de palabras y generadas por una determinada palabra x_i. A continuación, se decide cuáles son las palabras generadas por cada x_i. En función del número de palabras generadas, se decide cuántas palabras espúreas deben aparecer en la frase de salida y cuáles son (estas palabras se puede considerar que son generadas por la palabra vacía). Después, se asigna a cada una de las palabras generadas por las distintas x_i una posición en la frase $\bar{y}$. Por último, las palabras espúreas son colocadas en los huecos que han quedado libres en la frase $\bar{y}$.

Veamos cual sería el proceso de generación de la traducción de la frase $\bar{x} =$«*quiero una habitación doble .*», utilizando este modelo generativo. En primer

lugar, se decide la fertilidad de cada una de las palabras de la frase $\bar{x}$. Supongamos que optamos por que cada una de las palabras x_i posea una fertilidad $\phi_i = 1$, salvo *quiero* que tenga una fertilidad $\phi_1 = 2$. A continuación, dadas estas fertilidades, debemos decidir para cada palabra x_i qué palabras genera (en este caso qué palabra genera cada x_i). Supongamos que *quiero* genera las palabras *want* y *I* (no importa el orden, es una lista de palabras), *una* la palabra *a*, *habitación* la palabra *room* y así sucesivamente. Sabiendo que el número de palabras generadas es 5, tenemos que elegir ahora cuantas palabras espúreas deben aparecer en la frase $\bar{y}$ y cuáles son. Podría ser que optáramos porque en este caso no hubiera ninguna palabra espúrea. En caso contrario, tendríamos, además que decidir cuales son estas palabras. Por lo tanto, y en función de las anteriores elecciones tendremos 6 huecos en la frase $\bar{y}$ en los que deberemos distribuir las palabras seleccionadas:

$\bar{y} =$____, ____, ____, ____, ____, ____,

En primer lugar se ha decidir en qué huecos se colocan las palabras generadas por las distintas x_i. Supongamos que las palabras {*want*, *I*, *a*, *room*, *double*, «.»} acaben ocupando las siguientes posiciones:

$\bar{y} =$*I want a double room .*

Por último, se han de colocar en los huecos que quedasen las palabras espúreas. En este caso, puesto que no hay palabras espúreas ya habríamos terminado y la traducción finalmente generada sería:

$\bar{y} =$*I want a double room .*

Pese a que los tres modelos utilizan el mismo modelo generativo hay diferencias en los parámetros usados por cada uno de ellos. El modelo IBMm4 se diferencia del IBMm3 en que tiene en cuenta el hecho de que a menudo determinadas partes de una frase se traducen como una unidad. El modelo IBMm3 utiliza un parámetro al que denominan *distorsión* que determina la posición de cada palabra de forma independiente de la del resto de palabras, por lo que no tiene en cuenta la existencia de estas partes que han de moverse como una unidad. En el modelo IBMm4 se reemplaza dicho parámetro por dos nuevos parámetros: uno para determinar la posición del comienzo de cada unidad y otro para la posición del resto de palabras de dicha unidad. Para la determinación de dichos parámetros se utiliza un agrupamiento monolingüe en ambos idiomas.

Los modelos IBMm3 e IBMm4 son deficientes en tanto en cuanto desperdician parte de la probabilidad en frases mal formadas. IBMm5 es una reformulación de IBMm4 con un modelo de alineamiento adecuadamente refinado para evitar esta deficiencia.

2.5. Transductores subsecuenciales

El modelo de traducción utilizado en esta tesis, que pretende mejorarse mediante el uso del agrupamiento bilingüe y la detección de colocaciones, es el

modelo de transductores subsecuenciales.

Los transductores subsecuenciales (SSTs) son un tipo de modelos de estados finitos que han sido utilizados con éxito para pequeñas y medianas tareas de traducción ([ABC+97b], [VCJ+95]). La utilización de dichos traductores presenta una serie de ventajas. Puesto que se trata de modelos de autómatas finitos es sencillo integrarlos con reconocedores de voz. Son lo suficientemente simples como para que puedan ser aprendidos a partir de ejemplos lo suficientemente grandes. Además, son lo suficientemente potentes como para capturar varios de los fenómenos que se dan en la traducción entre dos idiomas, tal y como la diferente ordenación de las palabras que presentan muchos pares de lenguajes indoeuropeos.

Básicamente, un transductor subsecuencial (SST) [Ber79] es un autómata de estados finitos *determinista* que acepta frases en un determinado lenguaje y produce frases asociadas en un lenguaje de salida. Está formado por estados y transiciones que los unen. Cada transición tiene asociada un símbolo de entrada y una frase de salida. Que el autómata sea determinista implica que no puede haber distintas transiciones que partan de un mismo estado dado un mismo símbolo de entrada. El procesamiento de una frase de entrada parte de un estado diferenciado (denominado estado inicial) y procede consumiendo los símbolos de la frase de entrada uno a uno. Cada vez que un símbolo es aceptado, la frase asociada a la correspondiente transición es generada y se alcanza un nuevo estado. Este proceso continúa hasta que toda la entrada ha sido procesada; entonces, el último estado al que se ha llegado tras analizar la frase de entrada puede generar salida adicional.

Las posibles diferencias en la forma en la que dos idiomas ordenan las palabras se solventa permitiendo que el SST sea capaz de generar la cadena vacía. Por ejemplo, la frase de entrada *habitación doble* se traduciría por *double room* en dos pasos: después de haber visto la palabra *habitación* se genera la cadena vacía; y después de ver *doble* se generaría la frase *double room*. Las diferencias estructurales entre dos lenguajes se resuelven de forma similar. Por ejemplo, para una frase como *me llamo Antonio* el SST produce la cadena vacía después de haber visto *me*; luego después de haber visto *llamo* genera *my name is*; y finalmente *Antonio* genera *Antonio*.

Una descripción más detallada de los traductores subsecuenciales puede encontrarse en [Ber79].

Una de las ventajas diferenciadoras de los SSTs es el hecho de que pueden entrenarse de forma eficiente utilizando conjuntos de entrenamiento no ambiguos[2] con ejemplos de entradas y salidas. Este entrenamiento puede realizase utilizando OSTIA o algoritmos similares [Vil98]. Estos algoritmos realizan el entrenamiento básicamente en tres pasos:

- Se construye un árbol aceptor de prefijos utilizando las frases de entrada.

[2] El conjunto de entrenamiento no posee ninguna frase de entrada con dos traducciones distintas.

En dicho árbol, como subcadena de salida de todas las transiciones se asigna la cadena vacía; y cada frase de salida se asigna completamente al estado al que llega la correspondiente frase de entrada.

- Los prefijos comunes más largos de las cadenas de salida se mueven de forma recursiva, nivel a nivel, desde las hojas del árbol hacia su raíz.
- Comenzando por el estado raíz, todos los pares de estados se consideran de forma ordenada, nivel a nivel, y son mezclados si se considera aceptable que así sea: el traductor resultante es subsecuencial; no está en contradicción con el conjunto de entrenamiento; y se satisfacen una serie de restricciones.

Las restricciones mencionadas pueden ser de dos tipos: modelos de estados finitos para los lenguajes de entrada y salida, o bien diccionarios bilingües junto con alineamientos entre palabras. Para el primer caso, conviene utilizar una versión del algoritmo OSTIA, llamada OSTIA-DR; y en el segundo, la opción más adecuada es OMEGA (que además también puede tomar en cuenta modelos de lenguaje de la entrada y la salida). Los experimentos realizados en esta tesis utilizan el algoritmo OMEGA para la inferencia de los SSTs necesarios.

2.6. Categorías y transductores subsecuenciales

La calidad de los sistemas de traducción automática basados en SSTs puede mejorarse si se utilizan categorías en su entrenamiento. Los pasos necesarios para la generación de un SST que contenga toda la información necesaria para la traducción, incluyendo la de transductores específicos para cada categoría son [ABC+97a]:

- Identificación de las categorías. A modo de ejemplo, las categorías que se identificaron como más adecuadas en [ABC+97a] para la tarea en cuestión fueron: nombres propios masculinos, nombres propios femeninos, apellidos, fechas, números de habitación y números en general.
- Categorización del corpus. Una vez se han definido las categorías, se puede sustituir cada aparición de las palabras de una categoría por una etiqueta que identifique la categoría a la que pertenece. De esta forma, el par de frases *déme la llave de la habitación ciento veintitrés – give me the key to room one two three* se convertiría en *déme la llave de la habitación $ROOM – give me the key to room number $ROOM*; donde *$ROOM* sería la etiqueta asociada a la categoría de números de habitación.
- Aprendizaje del modelo inicial. A partir del corpus categorizado se infiere un SST inicial.
- Modelado de las categorías. Para cada categoría, se construye un SST sencillo: el SST de dicha categoría (cSST).

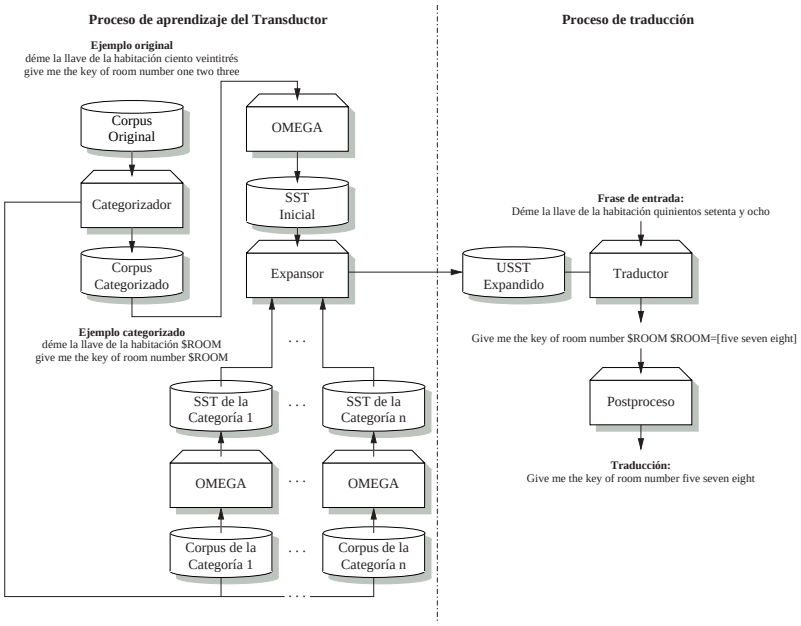


Figura 2.2: Esquema general de los procesos de aprendizaje y traducción utilizando categorías

- Expansión de las categorías. Los arcos en el SST inicial correspondientes a las diferentes categorías se expanden utilizando sus cSSTs.

El esquema general de los procesos de aprendizaje y traducción utilizando categorías se muestra en la Figura 2.2 en la que el algoritmo representado para la inferencia de los distintos SSTs es OMEGA.

Puesto que utilizando categorías definidas a mano se ha podido mejorar la traducción automática generada mediante SSTs, ¿podrían obtenerse clases bilingües de forma automática que mejoraran la traducción de SSTs? Esta tesis parte de dicha pregunta.

Capítulo 3

Agrupamiento monolingüe

3.1. Introducción

Como se ha visto en el capítulo anterior, el objetivo de un modelo estocástico de lenguaje es el de proporcionar una estimación de la probabilidad $\mathrm{P}(w_1 \dots w_N)$ de una secuencia de palabras $w_1 \dots w_N$. Esta probabilidad conjunta se calcula habitualmente como el producto de la secuencia de probabilidades condicionales $\mathrm{P}(w_i \mid w_1 \dots w_{i-1})$. Dicho producto, en el caso de utilizar un modelo de lenguaje de bigramas, queda reducido a $\prod_{i=1}^{N} \mathrm{P}(w_i \mid w_{i-1})$.

Uno de los principales problemas que presenta la estimación de un modelo de lenguaje es la escasez de los datos disponibles para su entrenamiento. Esto es así aún para el modelo relativamente sencillo de bigramas. Existen tantas posibles combinaciones de pares de palabras que es imposible observarlas todas el suficiente número de veces, por muy grande que sea el número de ejemplos del que dispongamos.

El agrupamiento de aquellas palabras que son similares entre sí es una forma de combatir la escasez de datos de entrenamiento disponibles. Si fuéramos capaces de agrupar satisfactoriamente palabras similares, podríamos realizar predicciones más razonables de aquellas secuencias de palabras que no se han visto asumiendo que son similares a otras ya vistas [BDd+92].

Es decir, un modelo de lenguaje puede utilizar las relaciones existentes entre grupos de palabras o categorías en lugar de basarse en las relaciones entre palabras individuales. Este enfoque presenta las siguientes ventajas [Nie97]:

- Los modelos basados en categorías comparten estadísticas entre palabras de la misma categoría y, por lo tanto, pueden generalizarlas a pautas de

palabras no encontradas en el corpus de entrenamiento. Esta habilidad de procesar los eventos no vistos de forma sensata mejora la *robustez* del modelo de lenguaje.

- Agrupar palabras en categorías puede reducir el número de contextos que debe considerar un modelo y, de ese modo, hacer frente a la escasez de datos del conjunto de entrenamiento.

- Además, la reducción en el número de contextos conlleva un modelo con menos parámetros y, por lo tanto, más compacto. Este modelo poseerá unos requerimientos de almacenamiento más modestos; lo que desde un punto de vista práctico es importante.

Por medio del agrupamiento se intenta aprovechar el hecho de que algunas palabras son similares entre sí, ya sea por su significado o por su función sintáctica. Por ejemplo, no nos resultaría sorprendente constatar que la distribución de probabilidad de las palabras próximas a *sábado* fuera similar a la de las palabras próximas a *martes*. Naturalmente, dichas distribuciones no tienen por qué ser idénticas: por ejemplo, difícilmente oiremos la frase *¡Gracias a Dios que ya es martes!* o a alguien preocuparse porque mañana sea *sábado 13* (mientras que si alguien es supersticioso puede preocuparle que mañana sea martes 13). Si hemos agrupado los días de la semana en una clase, podríamos suponer que en la mayor parte de contextos en los que aparece uno de los días de la semana podría aparecer cualquiera de los otros.

Es decir, el agrupamiento nos ayuda a generalizar; podríamos considerarlo como una forma de aprendizaje. Agrupamos palabras en clases y generalizamos lo que sabemos de algunos de los miembros de una clase a los restantes.

En este capítulo presentamos varios algoritmos que agrupan de forma automática palabras en clases. La siguiente sección muestra algunos conceptos generales de agrupamiento, así como los tipos de agrupamientos existentes y los algoritmos de agrupamiento monolingüe en los que se basan los propuestos en esta tesis. La sección 3.3 presenta la función que se quiere optimizar por medio del agrupamiento y que será utilizada por el algoritmo iterativo descrito en la sección 3.4. La sección 3.5 muestra un algoritmo idéntico al anterior pero en el que la función objetivo se obtiene utilizando un caso particular de validación cruzada. Por último, la sección 3.6 muestra un algoritmo que utiliza cualquiera de los dos anteriores para la obtención de un agrupamiento de forma incremental.

Las técnicas de agrupamiento descritas en este capítulo contemplan únicamente el caso de las probabilidades condicionales de bigramas $\mathrm{P}(w_i \mid w_{i-1})$; sin embargo, los conceptos presentados pueden ser fácilmente extendidos a trigramas o n-gramas en general.

3.2. Conceptos generales de agrupamiento

A lo largo de los años se han propuesto varias definiciones posibles del término agrupamiento (*clustering*); todas ellas generalmente basadas en la definición del término grupo o clase (*cluster*). Sin embargo, muchas de estas definiciones utilizan conceptos definidos pobremente tales como: «parecidos», «similares»... o por el contrario se refieren a grupos de un tipo determinado por lo que pierden su generalidad. Es más, muchas de las definiciones existentes son vagas y de tipo circular. Esto nos da una idea de la dificultad de encontrar una definición universalmente aceptada del término agrupamiento. La definición que hemos escogido, propuesta en [TK99, pp. 356–357], y que se presenta a continuación, responde al concepto de agrupamiento utilizado en esta tesis.

Sea X nuestro conjunto de datos,

$$X = \{x_1, x_2, \ldots, x_N\}\,. \tag{3.1}$$

Se define un m-agrupamiento de X, al que llamamos $\mathcal{G}$, como una partición de X en m conjuntos (*grupos* o *clases*), $c_1, \ldots, c_m$, de forma que se cumplan las siguientes condiciones:

- $c_i \neq \emptyset,\ i = 1, \ldots, m$,
- $\bigcup_{i=1}^{m} c_i = X$,
- $c_i \cap c_j = \emptyset,\ i \neq j,\ i,\ j = 1, \ldots, m$.

Además, buscaremos que los elementos de un grupo c_i sean «más similares» entre sí y «menos similares» a los elementos de los otros grupos. La cuantificación de los términos «similar» y «diferente» dependerá del tipo de grupos del que se hable.

Es conveniente notar que, de acuerdo a la anterior definición de agrupamiento, cada elemento pertenece a un sólo grupo. Este tipo de agrupamiento recibe el nombre de agrupamiento nítido[1] (*hard clustering* o *crisp clustering*).

Si quisiéramos encontrar el mejor agrupamiento posible, parece claro que, disponiendo del tiempo y recursos necesarios, la mejor forma de asignar cada elemento x_i a su grupo sería la de identificar todas las particiones posibles y seleccionar la más «razonable» de acuerdo a un criterio prefijado. Sin embargo, esta aproximación no es factible ni siquiera para un número moderado de elementos. Esto es debido a que el número de particiones distintas de N elementos en m grupos, $S(N, m)$, es [TK99, pp. 384]:

$$S(N, m) = \frac{1}{m!} \sum_{i=0}^{m} (-1)^{m-i} \left(\begin{array}{c} m \\ i \end{array} \right) i^N. \tag{3.2}$$

[1] Utilizamos el término *nítido* por oposición al término *difuso* y por ser traducción tanto de *hard* como de *crisp*; generalmente en castellano cuando se habla de agrupamiento se sobreentiende *hard-clustering* y cuando se quiere indicar *fuzzy clustering* se acompaña el término agrupamiento del adjetivo *difuso*. En [TK99, p. 357] se puede encontrar una definición de agrupamiento en términos de conjuntos difusos (*fuzzy sets*).

Dando valores a N y a m podemos obtener los siguientes resultados: $S(15,3) = 2.375.101$, $S(20,4) = 45.232.115.901$. Es decir, incluso para un número reducido de elementos, 15 y 20, agrupados en pocos grupos, 3 y 4, el número de particiones posibles es muy alto. Si aplicamos (3.2) para el número de elementos que querríamos agrupar en el caso de agrupamientos de palabras e intentamos agrupar 500 palabras (un vocabulario reducido) en tan sólo 2 grupos, el número de particiones que tendríamos que evaluar sería $S(500,2) \simeq 10^{150}$. (Si la evaluación de cada posible agrupamiento llevara tan sólo 10^{-20} segundos, un programa que evaluara todas las posibles agrupaciones de 500 palabras en 2 grupos tardaría más de 10^{122} años.)

Como se puede ver, la búsqueda de una solución óptima explorando todas las posibles agrupaciones es computacionalmente inviable. De hecho, incluso el problema más reducido de encontrar el agrupamiento óptimo por medio de una enumeración completa pero no explícita utilizando técnicas de ramificación y poda o programación dinámica constituyen problemas NP-duros [Bru78, JC99].

Vista la imposibilidad de evaluar todas las posibles particiones para encontrar el agrupamiento óptimo, los algoritmos de agrupamiento generalmente utilizados proporcionan agrupamientos razonables, subóptimos, considerando sólo una pequeña fracción del conjunto de todas las particiones posibles de X. Puesto que la solución no es necesariamente la óptima, los resultados obtenidos dependerán de cuál sea el criterio utilizado para la selección de este subconjunto.

Aunque existen una gran cantidad de algoritmos de agrupamiento, éstos pueden clasificarse en unos pocos tipos. Podemos distinguir en primer lugar, entre agrupamientos jerárquicos y no jerárquicos o planos dependiendo del tipo de estructura del agrupamiento generado [MS00]. Un agrupamiento jerárquico se puede ver como un árbol donde cada nodo representa una subclase de su nodo padre. Las hojas del árbol son los objetos del conjunto a agrupar (en nuestro caso las palabras). Cada nodo representa un grupo formado por todos los objetos de sus descendientes. Dichas estructuras se representan generalmente utilizando dendogramas. A modo de ejemplo, la figura 3.1 representa mediante un dendograma el agrupamiento jerárquico de 22 palabras inglesas frecuentes.

Los algoritmos jerárquicos pueden subdividirse a su vez en [TK99, pág. 385]:

- Aglomerativos. Producen una secuencia de agrupamientos con un número cada vez menor de grupos. El agrupamiento producido a cada paso se consigue juntando en un grupo, dos de los grupos presentes en el paso anterior.

- Divisivos. Actúan en la dirección contraria a los anteriores; esto es, producen una secuencia de agrupamientos con un número creciente de grupos a cada paso. El agrupamiento producido a cada paso es el resultado de dividir en dos, uno de los grupos presentes en el paso anterior.

Por otro lado, los agrupamientos planos están formado por un número fijo de clases sin relación alguna entre ellas. Muchos de los algoritmos que generan

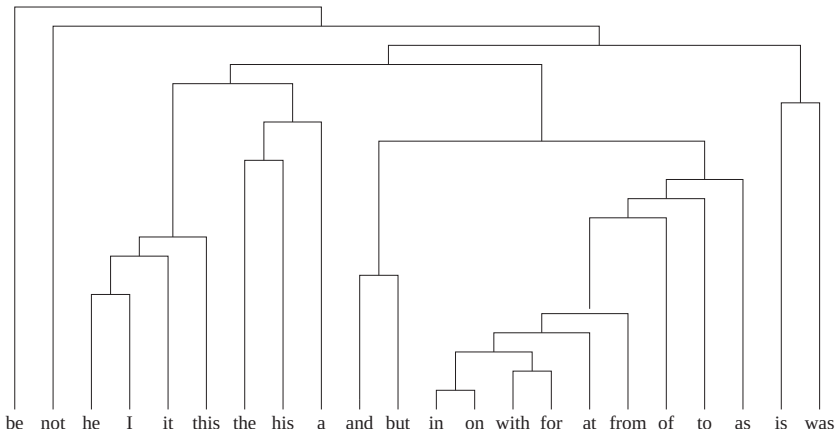


Figura 3.1: Dendograma que representa el agrupamiento jerárquico de 22 palabras inglesas frecuentes. Fuente: [MS00, pág. 496].

agrupamientos planos son *iterativos*. Comienzan con una distribución inicial de elementos en grupos y mejoran el agrupamiento mediante la recolocación iterativa de dichos elementos.

Otra distinción importante entre algoritmos de agrupamiento es si, como se ha visto antes, realizan un agrupamiento nítido o un agrupamiento difuso. En el agrupamiento nítido, cada objeto es asignado a un único grupo. Mientras que en el agrupamiento difuso se permite un cierto grado en la relación de pertenencia. Cada objeto puede pertenecer a más de un grupo. Dicha relación de pertenencia, puede expresarse en un marco probabilístico de la siguiente forma: para cada objeto x, se define una distribución de probabilidad $\mathrm{P}(\cdot \mid x)$ sobre los grupos, de tal forma que $\mathrm{P}(c \mid x)$ es la probabilidad de que x sea un miembro de c.

A pesar de su nombre, muchos de los algoritmos difusos asumen que un objeto pertenece en realidad a un sólo grupo, diferenciándose de los algoritmos nítidos en que existe una incertidumbre sobre cuál de los grupos posibles de un elemento es el correcto. También existen algoritmos difusos que sí que realizan asignación múltiple y donde cada elemento pertenece realmente a varios grupos. Dichos modelos reciben el nombre de agrupamientos disyuntivos [MS00].

Entre los agrupamientos jerárquicos, predominan los que asignan cada elemento a un único grupo. Entre los agrupamientos planos, los dos tipos de asignación son habituales.

Las principales características que presentan los algoritmos de agrupamiento jerárquicos y planos se muestran en el cuadro 3.1.

En cuanto al agrupamiento aplicado al campo del modelado de lenguaje natural, dos de los métodos de agrupamiento más conocidos son los propuestos por Brown et al. [BDd$^+$92]. En dicho trabajo se asume un agrupamiento nítido

Agrupamiento jerárquico:	Agrupamiento plano:
▪ Preferible para realizar análisis detallado de los datos. ▪ Proporciona mayor información que los agrupamientos planos. ▪ No existe un algoritmo mejor (varios algoritmos son óptimos para sus respectivas aplicaciones). ▪ Menos eficiente que el agrupamiento plano.	▪ Preferible si la eficiencia es importante o cuando los conjuntos de datos son muy grandes. ▪ El algoritmo de K-medianas es el método conceptualmente más simple y debería utilizarse en primer lugar sobre un conjunto de datos nuevos debido a que los resultados obtenidos son a menudo suficientes.

Cuadro 3.1: Resumen de las principales características de los algoritmos de agrupamiento.

en el que cada palabra w forma parte de sólo una clase $\mathcal{G}(w)$. Se propone la estimación basada en clases de la probabilidad de w_2 dada w_1 como:

$$\mathrm{P}(w_2 \mid w_1) = \mathrm{P}(w_2 \mid \mathcal{G}(w_2))\mathrm{P}(\mathcal{G}(w_2) \mid \mathcal{G}(w_1)). \tag{3.3}$$

Dado un corpus de entrenamiento y la función $\mathcal{G}$, tanto $\mathrm{P}(w \mid \mathcal{G}(w))$ como $\mathrm{P}(\mathcal{G}(w_2) \mid \mathcal{G}(w_1))$ pueden determinarse contando el número de veces que se dan en el entrenamiento determinados eventos. Por lo tanto, sólo es necesario encontrar la función $\mathcal{G}$. Brown et al. [BDd+92] proponen encontrar esta función $\mathcal{G}$ de forma que la perplejidad del modelo de lenguaje basado en clases se reduzca; lo que equivale a maximizar la información mutua entre clases adyacentes $\mathrm{I}(c_1; c_2)$ (ver sección 3.3).

El primero de los algoritmos propuestos por Brown et al. [BDd+92] es un algoritmo jerárquico aglomerativo en el que parten de una distribución inicial en la que hay tantas clases como palabras y van juntando dos clases cada vez hasta que se alcanza el número de clases deseado. En cada una de las iteraciones se prueban todos los posibles agrupamientos hasta que se encuentra aquel que produce el mayor aumento en $\mathrm{I}(c_1; c_2)$. Además, para compensar posibles agrupamientos prematuros de palabras en una misma clase, se desplazan palabras de unas clases a otras una vez se ha alcanzado el número deseado de clases. El algoritmo propuesto presenta un coste temporal $O(V^3)$, donde V es la talla del vocabulario.

Para disminuir dicho coste, Brown et al. [BDd+92] presentan otro algoritmo que reduce la complejidad temporal a $O(V \cdot C^2)$, siendo C el número de clases deseadas. Este algoritmo ordena las palabras por su frecuencia y coloca las C primeras en una clase cada una. En cada iteración se añade la siguiente palabra más frecuente aún no agrupada como una nueva clase, buscando entonces qué dos clases es mejor unir. Cuando se ha realizado la fusión, el sistema vuelve a tener C clases. Pese a la reducción en coste temporal obtenida, es posible que este

heurístico acote tanto la búsqueda que muchos buenos agrupamientos no sean tenidos en cuenta [Lee97].

Otro de los métodos de agrupamiento monolingüe ampliamente citado, es el de Kneser y Ney [KN93]. En muchos aspectos, el trabajo de Kneser y Ney es bastante similar al de Brown et al. Utilizan el mismo modelo de probabilidad condicional basada en clases, y algunos de los heurísticos para acelerar los cálculos son también idénticos. Sin embargo, el criterio de optimización difiere, a pesar de que también deriva del principio de máxima-verosimilitud. La diferencia más importante es que en lugar de utilizar un algoritmo aglomerativo, Kneser y Ney [KN93] utilizan un algoritmo de agrupamiento plano que, por lo tanto, tiene desde el principio el número deseado de clases. La operación básica que realizan para buscar un mejor agrupamiento es el movimiento de una palabra desde su clase actual a otra. El coste temporal de dicho algoritmo para cada iteración es $O(N + V \cdot C^2)$, donde N es el número de palabras del corpus de entrenamiento, V la talla del vocabulario y C el número de clases deseadas.

Martin et al. [MLN95] mejoran el método propuesto por Kneser y Ney, organizando el corpus de entrenamiento de la siguiente forma: para cada par de palabras observadas, almacenan su cuenta. De esta forma, en lugar de visitar cada ocurrencia de la palabra w en el corpus de entrenamiento, se recopilan todos aquellos bigramas en los que aparece w. Asumiendo que dichas cuentas se almacenan en una matriz de acceso directo, obtienen una complejidad temporal para cada iteración de: $O(B + V \cdot C^2)$, donde B es el número de bigramas vistos, que habitualmente es bastante menor que el número de palabras de entrenamiento.

Para vocabularios grandes, el acceso directo se vuelve prohibitivo debido a los grandes requerimientos de memoria. Por ello, Martin et al. [MLN95] proponen utilizar listas y búsqueda binaria para almacenar las cuentas de bigramas. Puesto que por cada palabra hay de media B/V bigramas, su algoritmo presenta la siguiente complejidad temporal: $O(B \cdot \log(\frac{B}{V}) + V \cdot C^2)$.

Los algoritmos de agrupamiento monolingüe presentados en esta tesis están basados en los algoritmos de Brown et al. [BDd+92], Kneser y Ney [KN93] y Martin et al. [MLN95].

3.3. Función objetivo

El agrupamiento de palabras que queremos realizar tiene por objeto la obtención de un modelo de lenguaje mejor. Por lo tanto, es conveniente conocer la relación existente entre un agrupamiento concreto y el correspondiente modelo de lenguaje basado en clases. Conociendo esta relación, podremos buscar aquel agrupamiento que proporcione el mejor modelo de lenguaje basado en clases. En esta sección presentamos la función objetivo que utilizaremos en los algoritmos de agrupamiento propuestos.

Aplicando el modelo de bigramas, en el cual una palabra depende únicamente de la palabra anterior, podemos aproximar la entropía cruzada de un corpus

$\mathrm{L} = w_1 \dots w_N$ para una función $\mathcal{G}$ de asignación de palabras a grupos como:

$$\begin{aligned}
\mathrm{H}(\mathrm{L}, \mathcal{G}) &= -\frac{1}{N} \log \mathrm{P}_G(w_1 \dots w_N) \\
&\simeq -\frac{1}{N-1} \log \prod_{i=2}^{N} \mathrm{P}_G(w_i \mid w_{i-1}) \\
&\simeq -\frac{1}{N-1} \sum_{vw} \mathrm{n}(vw) \log \mathrm{P}_G(w \mid v). \qquad (3.4)
\end{aligned}$$

Si consideramos la función $\mathcal{G}$ como una función que, dada una palabra w, proporciona la clase c a la que ha sido asignada, no pudiendo pertenecer una palabra a más de una clase, podemos aproximar $\mathrm{P}_G(w \mid v)$ como:

$$\mathrm{P}_G(w \mid v) \simeq \mathrm{P}(w \mid c_2)\mathrm{P}(c_2 \mid c_1), \qquad (3.5)$$

donde $c_1 = \mathcal{G}(v)$ y $c_2 = \mathcal{G}(w)$. Ésta es la formula utilizada en Brown et al. [BDd$^+$92] y ya presentada en (3.3).

Es decir, la probabilidad de que veamos la palabra w si hemos visto la palabra v viene dada mediante la probabilidad de w sabiendo su clase c_2 por la probabilidad de ver la clase c_2 una vez vista la clase c_1 (a la que pertenece la palabra v).

Sustituyendo (3.5) en (3.4), $\mathrm{H}(\mathrm{L}, \mathcal{G})$ queda:

$$\mathrm{H}(\mathrm{L}, \mathcal{G}) \simeq -\frac{1}{N-1} \sum_{vw} \mathrm{n}(vw) \log \mathrm{P}(w \mid c_2)\mathrm{P}(c_2 \mid c_1). \qquad (3.6)$$

Si realizamos las siguiente serie de operaciones [MS00, pp. 510–511]:

$$\begin{aligned}
\mathrm{H}(\mathrm{L}, \mathcal{G}) &\simeq -\Bigg[\sum_{uw} \frac{\mathrm{n}(uw)}{N-1} \Big(\log \mathrm{P}(w \mid c_2) + \log \mathrm{P}(c_2)\Big) \\
&\qquad + \sum_{uw} \frac{\mathrm{n}(uw)}{N-1} \Big(\log \mathrm{P}(c_2 \mid c_1) - \log \mathrm{P}(c_2)\Big)\Bigg] \\
&\simeq -\Bigg[\sum_{w} \frac{\sum_u \mathrm{n}(uw)}{N-1} \log \mathrm{P}(w \mid c_2)\mathrm{P}(c_2) \\
&\qquad + \sum_{c_1 c_2} \frac{\mathrm{n}(c_1 c_2)}{N-1} \log \frac{\mathrm{P}(c_2 \mid c_1)}{\mathrm{P}(c_2)}\Bigg] \\
&\simeq -\sum_{w} \mathrm{P}(w) \log \mathrm{P}(w) - \sum_{c_1 c_2} \mathrm{P}(c_1 c_2) \log \frac{\mathrm{P}(c_1 c_2)}{\mathrm{P}(c_1)\mathrm{P}(c_2)} \\
&\simeq \mathrm{H}(w) - \mathrm{I}(c_1; c_2), \qquad (3.7)
\end{aligned}$$

llegamos a que $\mathrm{H}(\mathrm{L},\mathcal{G}) \simeq \mathrm{H}(w) - \mathrm{I}(c_1;c_2)$. Es decir, la entropía cruzada puede calcularse como la entropía del corpus dada la probabilidad de las palabras que lo forman, $\mathrm{H}(w)$, menos la información mutua entre clases adyacentes, $\mathrm{I}(c_1;c_2)$. Puesto que $\mathrm{H}(w)$ es independiente de la función $\mathcal{G}$ elegida, la función $\mathcal{G}_{opt}$ buscada será por tanto:

$$\mathcal{G}_{opt} = \arg\max_{\mathcal{G}} \mathrm{I}(c_1;c_2) = \arg\max_{\mathcal{G}} \sum_{c_1 c_2} \mathrm{P}(c_1 c_2) \log \frac{\mathrm{P}(c_1 c_2)}{\mathrm{P}(c_1)\mathrm{P}(c_2)}. \tag{3.8}$$

Para la obtención de un agrupamiento, basado en la optimización de la anterior función, se proponen en esta tesis los siguientes algoritmos:

- Algoritmo iterativo.
- Algoritmo iterativo dejando uno fuera (*leaving one out*).
- Algoritmo incremental.

El primero de ellos, el algoritmo iterativo, descrito en la siguiente sección, implementa un algoritmo similar al algoritmo de intercambio (*exchange algorithm*) utilizado en el agrupamiento convencional y busca mejorar $\mathrm{I}(c_1;c_2)$ sobre el corpus de entrenamiento.

El segundo, el algoritmo iterativo dejando uno fuera, descrito en la sección 3.5, es una variante del primero pero que utiliza la técnica de dejar uno fuera para el cálculo de $\mathrm{I}(c_1;c_2)$.

El último de ellos, el algoritmo incremental, descrito en la sección 3.6, comienza con una sola clase e incrementa una a una el número de clases hasta llegar al número deseado de clases; para cada número de clases utiliza alguno de los algoritmos anteriores para encontrar el mejor agrupamiento.

3.4. Algoritmo iterativo

El primero de los algoritmos presentados, al que llamamos algoritmo de agrupamiento iterativo, se basa en el algoritmo de agrupamiento de Kneser y Ney [KN93] junto con las mejoras de Martin et al. [MLN95] y una serie de consideraciones presentadas en esta sección. El criterio de optimización que hemos utilizado es el de la mejora de la perplejidad (ver sección 3.3).

Este algoritmo (ver algoritmo 3.1) genera una distribución inicial de palabras en clases y modifica esta distribución, moviendo iterativamente palabras de unas clases a otras, buscando aumentar la información mutua entre clases adyacentes, $\mathrm{I}(c_1;c_2)$.

La idea principal para cambiar la distribución es la siguiente: dada la función objetivo, $\mathrm{I}(c_1;c_2)$, que depende del agrupamiento, se calcula cómo se modificaría el valor de dicha función si una determinada palabra se moviera de la clase en la que se encuentra a cualquier otra; si alguno de los movimientos mejora $\mathrm{I}(c_1;c_2)$,

Algoritmo 3.1 Agrupamiento iterativo

Requiere: C, $maxIter$
  Inicializar $\mathcal{G}$
  repetir
    para cada w en orden descendiente de frecuencia **hacer**
      Mover tentativamente w a cada clase c
      Mover w a la clase c que más incremente $\mathrm{I}(c_1; c_2)$
    fin para
  hasta que se alcance $maxIter$ o no se haya movido ninguna palabra

se mueve definitivamente la palabra a aquella clase para la que se ha obtenido la mayor mejora. Este procedimiento se realiza para todas las palabras en un determinado orden y el proceso completo se repite hasta que ya no hay más movimientos o hasta que se alcanza un número prefijado de iteraciones.

Las palabras se evalúan en el siguiente orden: de las más frecuentes, es decir, las que aparecen más veces en el corpus, a las menos frecuentes. El utilizar dicho orden permite determinar primero la clase a la que pertenecen aquellas palabras de las que se posee una mayor evidencia en el corpus.

Además, otro aspecto a tener en cuenta es que el algoritmo propuesto generará un agrupamiento u otro dependiendo de cuál sea la distribución inicial de palabras en clases escogida. La distribución inicial propuesta en [BDd$^+$92] y en [MLN95] es la de colocar las palabras más frecuentes cada una en una clase y las restantes en la última clase. Esta distribución inicial, en la técnica propuesta en [BDd$^+$92] permite el agrupamiento de las palabras considerando en primer lugar a las palabras más frecuentes puesto que, como se comentó en la sección 3.2, esta técnica consiste en la asignación, en primer lugar, de las C palabras más frecuentes cada una a una clase y posteriormente en la creación de una nueva clase con la siguiente palabra y la unión de dos de estas clases en una sola, repitiendo estos dos pasos hasta que no queden más palabras. De esta forma, las palabras frecuentes son consideradas justo al comienzo del proceso y condicionan la distribución de palabras en clases.

Martin et al [MLN95] también proponen dicha distribución inicial. Sin embargo, en este caso el algoritmo impide que las palabras más frecuentes sean evaluadas al comienzo, ya que al estar cada una en una clase, no pueden abandonar dichas clases hasta que dejen de ser las únicas palabras en ellas. Una distribución inicial más adecuada, a nuestro juicio, sería la de colocar todas las palabras más frecuentes en una clase, y las $C - 1$ palabras menos frecuentes en las restantes $C - 1$ clases, ya que de esta forma, las palabras más frecuentes se mueven en primer lugar dando lugar a mejores agrupamientos [BV99c].

Partiendo de esta distribución inicial, las palabras se mueven de una clase a otra buscando mejorar la función objetivo $\mathrm{I}(c_1; c_2)$ (ver sección 3.3). Desde el punto de vista de la implementación del algoritmo conviene observar que

dicha función no tiene por qué recalcularse para cada movimiento si no que se puede calcular directamente el incremento o decremento, $\Delta \mathrm{I}(c_1; c_2)$, debido a cada movimiento de forma similar a la planteada en [MLN95].

Además, para reducir el tiempo de cómputo de los $\Delta \mathrm{I}(c_1; c_2)$ debidos al movimiento de una palabra w a cada una de las posibles clases $c_1, \ldots, c_C$ se han utilizado las siguientes técnicas:

- Se ha separado el cómputo del $\Delta \mathrm{I}(c_1; c_2)$ en dos partes: la debida a la extracción de w de c_w y la debida a la inserción de w en c_d. De esta forma, el $\Delta \mathrm{I}(c_1; c_2)$ debido a la extracción se calcula sólo una vez cuando se evalúan los posibles movimientos de la palabra w.
- Se limita el número de cuentas a recalcular obteniendo previamente aquellas clases que preceden y siguen a la palabra w. Como se verá más adelante estas clases son las únicas que intervienen en la modificación de $\mathrm{I}(c_1; c_2)$. Esto se hace una vez por palabra que se quiere evaluar.

Expresando $\mathrm{I}(c_1; c_2)$ en función de las cuentas sobre el corpus de entrenamiento, tenemos:

$$\mathrm{I}(c_1; c_2) \simeq \frac{1}{N-1} \left[\sum_{c_1 c_2} \mathrm{n}(c_1 c_2) \cdot \log \frac{\mathrm{n}(c_1 c_2)}{N-1} - 2 \sum_{c} \mathrm{n}(c) \cdot \log \frac{\mathrm{n}(c)}{N-1} \right]. \quad (3.9)$$

Podemos expresar la variación de $\mathrm{n}(c_1 c_2)$ y $\mathrm{n}(c)$ debida a la extracción de una palabra w de su clase c_w como:

$$\mathrm{n}'(c_1 c_2) = \begin{cases} \mathrm{n}(c_1 c_2) - \mathrm{n}(w c_2) & \text{si } c_1 = c_w \wedge c_2 \neq c_w \\ \mathrm{n}(c_1 c_2) - \mathrm{n}(c_1 w) & \text{si } c_1 \neq c_w \wedge c_2 = c_w \\ \mathrm{n}(c_1 c_2) - \mathrm{n}(c_w w) & \\ \quad - \mathrm{n}(w c_w) + \mathrm{n}(ww) & \text{si } c_1 = c_2 = c_w \\ \mathrm{n}(c_1 c_2) & \text{en otro caso} \end{cases} \quad (3.10)$$

$$\mathrm{n}'(c) = \begin{cases} \mathrm{n}(c) - \mathrm{n}(w) & \text{si } c = c_w \\ \mathrm{n}(c) & \text{en otro caso} \end{cases} \quad (3.11)$$

De igual forma, podemos calcular la variación de $\mathrm{n}(c_1 c_2)$ y $\mathrm{n}(c)$ si w se mueve a c_d como:

$$\mathrm{n} \gg (c_1 c_2) = \begin{cases} \mathrm{n}'(c_1 c_2) + \mathrm{n}(w c_2) & \text{si } c_1 = c_d \wedge c_2 \neq c_d \\ \mathrm{n}'(c_1 c_2) + \mathrm{n}(c_1 w) & \text{si } c_1 \neq c_d \wedge c_2 = c_d \\ \mathrm{n}'(c_1 c_2) + \mathrm{n}(ww) & \\ \quad + \mathrm{n}(w c_d) + \mathrm{n}(c_d w) & \text{si } c_1 = c_2 = c_d \\ \mathrm{n}'(c_1 c_2) & \text{en otro caso} \end{cases} \quad (3.12)$$

$$\mathrm{n} \gg (c) = \begin{cases} \mathrm{n}'(c) + \mathrm{n}(w) & \text{si } c = c_d \\ \mathrm{n}'(c) & \text{en otro caso} \end{cases} \quad (3.13)$$

En función de dichas cuentas, podemos calcular el $\Delta \mathrm{I}(c_1; c_2)$ debido a la extracción de la palabra w de la clase c_w como:

$$\begin{aligned}
\Delta \mathrm{I}(c_1; c_2)\,|_{ext} \quad \simeq \quad & \frac{1}{N-1}\left[\mathrm{n}'(c_w, c_w) \cdot \log \frac{\mathrm{n}'(c_w, c_w)}{N-1}\right. \\
& -\mathrm{n}(c_w, c_w) \cdot \log \frac{\mathrm{n}(c_w, c_w)}{N-1} \\
& + \sum_{c: c \neq c_w} \left(\mathrm{n}'(c, c_w) \cdot \log \frac{\mathrm{n}'(c, c_w)}{N-1} - \mathrm{n}(c, c_w) \cdot \log \frac{\mathrm{n}(c, c_w)}{N-1}\right) \\
& + \sum_{c: c \neq c_w} \left(\mathrm{n}'(c_w, c) \cdot \log \frac{\mathrm{n}'(c_w, c)}{N-1} - \mathrm{n}(c_w, c) \cdot \log \frac{\mathrm{n}(c_w, c)}{N-1}\right) \\
& \left. -2\left(\mathrm{n}'(c_w) \cdot \log \frac{\mathrm{n}'(c_w)}{N-1} - \mathrm{n}(c_w) \cdot \log \frac{\mathrm{n}(c_w)}{N-1}\right)\right]. \qquad (3.14)
\end{aligned}$$

Por otro lado, podemos calcular el $\Delta \mathrm{I}(c_1; c_2)$ debido a la inserción de la palabra w en la clase c_d como:

$$\begin{aligned}
\Delta \mathrm{I}(c_1; c_2)\,|_{ins} \quad \simeq \quad & \frac{1}{N-1}\left[\mathrm{n} \gg (c_d, c_d) \cdot \log \frac{\mathrm{n} \gg (c_d, c_d)}{N-1}\right. \\
& -\mathrm{n}(c_d, c_d) \cdot \log \frac{\mathrm{n}(c_d, c_d)}{N-1} \\
& + \sum_{c: c \neq c_d} \left(\mathrm{n} \gg (c, c_d) \cdot \log \frac{\mathrm{n} \gg (c, c_d)}{N-1} - \mathrm{n}'(c, c_d) \cdot \log \frac{\mathrm{n}'(c, c_d)}{N-1}\right) \\
& + \sum_{c: c \neq c_d} \left(\mathrm{n} \gg (c_d, c) \cdot \log \frac{\mathrm{n} \gg (c_d, c)}{N-1} - \mathrm{n}'(c_d, c) \cdot \log \frac{\mathrm{n}'(c_d, c)}{N-1}\right) \\
& \left. -2\left(\mathrm{n} \gg (c_d) \cdot \log \frac{\mathrm{n} \gg (c_d)}{N-1} - \mathrm{n}'(c_d) \cdot \log \frac{\mathrm{n}'(c_d)}{N-1}\right)\right]. \qquad (3.15)
\end{aligned}$$

Finalmente, obtenemos el $\Delta \mathrm{I}(c_1; c_2)$ debido al movimiento de la palabra w de la clase c_w a la clase c_d como:

$$\Delta \mathrm{I}(c_1; c_2) = \Delta \mathrm{I}(c_1; c_2)\,|_{ext} + \Delta \mathrm{I}(c_1; c_2)\,|_{ins}\ .$$

Si observamos las ecuaciones 3.10 y 3.12, podemos ver como efectivamente sólo hay variaciones en las cuentas en aquellos casos en los que o $\mathrm{n}(c, w) \neq 0$ o $\mathrm{n}(w, c) \neq 0$; es decir, en aquellos casos en los que una clase antecede o sigue a la palabra que se está evaluando. Por lo tanto, si cuando queremos evaluar el $\Delta \mathrm{I}(c_1; c_2)$ debido al movimiento de una palabra w obtenemos previamente las clases que la anteceden y la siguen, sólo tendremos que realizar las anteriores operaciones sobre estas clases. Ésta es la segunda de las técnicas de optimización propuestas anteriormente.

3.5. Algoritmo iterativo dejando uno fuera

Si observamos cómo varía con el número de clases la entropía medida sobre un conjunto de test, de un modelo de lenguaje basado en bigramas de clases, podemos comprobar como ésta inicialmente decae hasta alcanzar un mínimo, y como luego vuelve a ascender hasta llegar al valor de la entropía de un modelo de bigramas (cuando el número de clases es igual al del vocabulario). Este comportamiento se explica mediante las siguientes fases [Nie97]:

- Sobregeneralización. Cuando el número de clases es reducido, el modelo de lenguaje apenas puede distinguir entre secuencias de historias que realmente son diferentes entre sí.

- Compensación. Cuando el número de clases aumenta, el grado de generalización del modelo se reduce y la habilidad para discriminar entre diferentes historias mejora.

- Sobreentrenamiento. Cuando el número de clases es demasiado grande, el modelo de lenguaje comienza a reflejar las peculiaridades del conjunto de entrenamiento, por lo que es de esperar que se comporte peor sobre el conjunto de test.

Cuando la entropía se mide sobre el mismo conjunto de entrenamiento sobre el que se estiman las distintas probabilidades —y no sobre el conjunto de test—, la variación de ésta con el número de clases no sigue la pauta anterior. Puesto que los estimadores por máxima-verosimilitud de las probabilidades son las frecuencias relativas de los bigramas en el conjunto de entrenamiento, la entropía de un modelo de bigramas de clases medida sobre este conjunto de entrenamiento mejora conforme aumenta el número de clases, hasta que el número de clases es igual al del vocabulario; es decir, hasta que el modelo basado en clases equivale al modelo de bigramas. Por lo tanto, la entropía medida sobre el conjunto de entrenamiento es una función decreciente con el número de clases y no puede utilizarse para determinar el número óptimo de clases.

La figura 3.2 muestra la variación con el número de clases de las perplejidades medidas sobre el conjunto de test y sobre el conjunto de entrenamiento. Para la realización de este experimento se ha utilizado la parte en castellano del corpus castellano–inglés EuTrans I (ver capítulo 6) y los distintos agrupamientos se han obtenido con el algoritmo iterativo descrito en la sección anterior. En la figura se han representado: la perplejidad medida sobre el test para un modelo de bigramas y para modelos de bigramas de clases con distintos números de clases; y la perplejidad medida sobre el entrenamiento para un modelo de bigramas y para modelos de bigramas de clases con distintos números de clases. Si se observa la variación de la perplejidad medida sobre el test para los modelos de bigramas de clases para los distintos números de clases, se puede apreciar como ésta efectivamente disminuye al principio conforme el número de clases aumenta

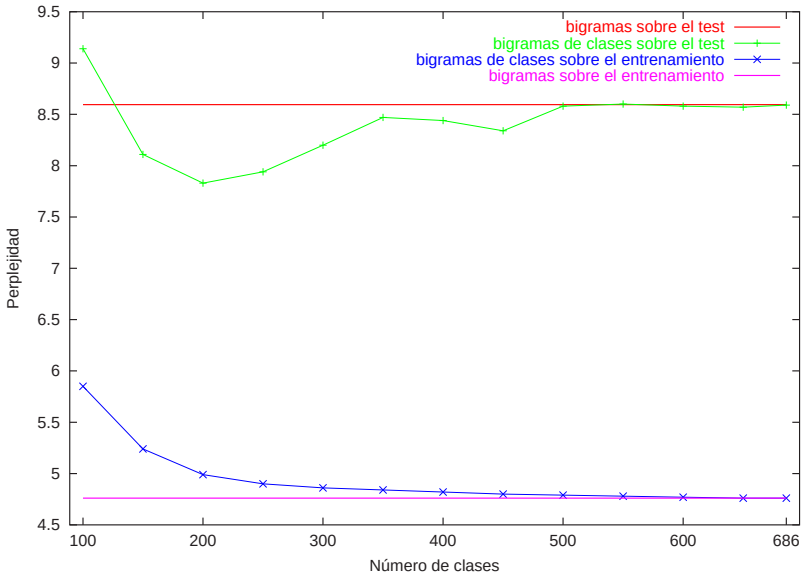


Figura 3.2: Perplejidad medida sobre el test y el entrenamiento para bigramas y bigramas de clases utilizando el algoritmo iterativo.

para después comenzar a ascender. Además, también se puede observar como la perplejidad medida sobre el conjunto de entrenamiento para los modelos de bigramas de clases disminuye conforme el número de clases aumenta y alcanza su valor mínimo cuando el número de clases es igual al del vocabulario del conjunto de entrenamiento (686 en el ejemplo).

Para evitar el sobreentrenamiento, se suele utilizar una técnica consistente en la división del corpus de entrenamiento en dos partes: a una de estas partes se le denomina la parte retenida (*retained part*) y a la otra, generalmente más pequeña, la parte separada (*held-out part*). Las estimaciones iniciales se llevan a cabo realizando las cuentas sobre la parte retenida y posteriormente se refinan sobre la parte separada. El coste que conlleva la utilización de esta técnica es la reducción de datos disponibles, siempre escasos, de entrenamiento, por lo que las estimaciones obtenidas pueden ser menos fiables [MS00].

Para solucionar el problema de la disminución de datos de entrenamiento, se aplican otros esquemas más eficientes en los que cada parte de los datos de entrenamiento se utiliza tanto para el entrenamiento inicial como para el posterior refinamiento. Por regla general, dichos métodos reciben en estadística el nombre de métodos de validación cruzada (*cross validation methods*).

El método de *dejar uno fuera* (*leaving-one-out method*) es un caso particular de validación cruzada. Este método divide el corpus de entrenamiento en una

parte retenida con $N-1$ muestras y en una parte separada con la muestra restante. El proceso se repite N veces de tal forma que se tienen en consideración todas las N particiones posibles con 1 muestra separada. La ventaja básica de esta aproximación es que todas las muestras son usadas tanto en la parte retenida como en la parte separada y por lo tanto se consigue aprovechar al máximo el corpus de entrenamiento disponible [KN93].

Utilizando este método para calcular la función objetivo en la que se basa el agrupamiento se puede evitar el sobreentrenamiento y obtener una estimación del número óptimo de clases.

Si llamamos T_i al corpus de entrenamiento sin el evento $w_{i-1}w_i$, la entropía cruzada del lenguaje según el modelo de lenguaje basado en clases (ecuación 3.4) se puede reescribir:

$$\mathrm{H}_{lo}(\mathrm{L},\mathcal{G}) \simeq -\frac{1}{N-1}\log\prod_{i=2}^{N}\mathrm{P}_{G,T_i}(w_i \mid w_{i-1}). \tag{3.16}$$

Si llamamos T_{vw} al corpus de entrenamiento sin uno de los eventos vw, tenemos:

$$\mathrm{H}_{lo}(\mathrm{L},\mathcal{G}) \simeq -\sum_{vw}\frac{\mathrm{n}(vw)}{N-1}\log \mathrm{P}_{G,T_{vw}}(c_2 \mid c_1)\mathrm{P}_{G,T_{vw}}(w \mid c_2), \tag{3.17}$$

donde, por simplificar la notación posterior, hemos denominado c_2 a $\mathcal{G}(w)$ y c_1 a $\mathcal{G}(v)$.

Tras realizar la serie de operaciones vistas en la sección 3.3, la parte de la ecuación 3.17 que depende del agrupamiento realizado queda:

$$\mathrm{I}_{lo}(c_1;c_2) \simeq \sum_{c_1c_2}\frac{\mathrm{n}(c_1c_2)}{N-1}\log \mathrm{P}_{G,T_{c_1c_2}}(c_1c_2) - 2\sum_{c}\frac{\mathrm{n}(c)}{N-1}\log \mathrm{P}_{G,T_c}(c), \tag{3.18}$$

donde abusando de la notación se ha denominado $T_{c_1c_2}$ al corpus de entrenamiento sin un evento vw cualquiera tal que $\mathcal{G}(v)=c_1 \wedge \mathcal{G}(w)=c_2$ y T_c al corpus de entrenamiento sin un evento vw cualquiera tal que $\mathcal{G}(v)=c \vee \mathcal{G}(w)=c$.

De ahora en adelante, se utilizará $\mathrm{P}_{G,T}$ para representar la probabilidad, dada $\mathcal{G}$, estimada sobre el corpus de entrenamiento del que se ha retirado un determinado evento vw.

Es decir, para cada evento vw presente en el corpus de entrenamiento queremos estimar $\mathrm{P}_{G,T}(c_1c_2)$ utilizando un corpus del que se ha eliminado dicho evento. Por lo tanto, es necesario considerar la posibilidad de que haya eventos no vistos en la parte retenida del corpus de entrenamiento a los que hay que asignar una probabilidad no nula. Para ello, siguiendo a Kneser y Ney [KN93], se utiliza el método de suavizado de descuento absoluto. Dicho método descuenta una cantidad $b<1$ a cada cuenta y reparte la masa de probabilidad ganada entre los eventos no vistos. Utilizando $n_{+,T}$ para representar el número de pares

c_1c_2 que se han visto en la parte retenida y $n_{0,T}$ para representar el número de los que no, se tiene que:

$$\mathrm{P}_{G,T}(c_1c_2) = \begin{cases} \dfrac{\mathrm{n}_T(c_1c_2) - b}{N_T} & \text{si } \mathrm{n}_T(c_1c_2) > 0 \\ \dfrac{n_{+,T}\, b}{n_{0,T} N_T} & \text{si } \mathrm{n}_T(c_1c_2) = 0 \end{cases} \tag{3.19}$$

donde $n_{+,T}$ y $n_{0,T}$ se definen formalmente como:

$$n_{+,T} = \sum_{c_1c_2 : \mathrm{n}_T(c_1c_2) \geq 1} 1 \quad \text{y} \quad n_{0,T} = \sum_{c_1c_2 : \mathrm{n}_T(c_1c_2) = 0} 1. \tag{3.20}$$

Es decir, los $n_{+,T}$ pares de clases que han sido vistos al menos una vez en la parte retenida del corpus proporcionan una masa de probabilidad $n_{+,T}\frac{b}{N_T}$ que se distribuye de forma uniforme entre los $n_{0,T}$ eventos no vistos en la parte retenida.

Eliminando un evento c_1c_2 del corpus, se reducirá en uno el número de ocurrencias de dicho evento en todo el corpus. Por lo tanto, si eliminamos un evento que ocurre exactamente una vez, se incrementará el número n_0 de bigramas no vistos en todo el corpus y se decrementará el número n_+ de bigramas visto alguna vez en todo el corpus.

Es posible expresar $\mathrm{P}_{G,T}(c_1c_2)$ en función de las cuentas $\mathrm{n}(\cdot)$, n_0 y n_+ sobre el corpus de entrenamiento completo, análogas a $\mathrm{n}_T(\cdot)$, $n_{0,T}$ y $n_{+,T}$ sobre la parte retenida, como:

$$\mathrm{P}_{G,T}(c_1c_2) = \begin{cases} \dfrac{\mathrm{n}(c_1c_2) - 1 - b}{N - 2} & \text{si } \mathrm{n}(c_1c_2) > 1 \\ \dfrac{(n_+ - 1)b}{(n_0 + 1)(N - 2)} & \text{si } \mathrm{n}(c_1c_2) = 1 \end{cases} \tag{3.21}$$

Por otro lado, para la estimación de $\mathrm{P}_{G,T_c}(c)$, se pueden utilizar las cuentas relativas siempre que se garantice que cada clase ocurre al menos una vez en la parte retenida del corpus. Dicha estimación sería:

$$\mathrm{P}_{G,T_c}(c) = \frac{\mathrm{n}_T(c)}{N_T} = \frac{\mathrm{n}(c) - 1}{N - 2}. \tag{3.22}$$

Una forma de garantizar que cada clase ocurre al menos una vez en el corpus retenido es evitando explícitamente que se pueda mover una determinada palabra w_i de su clase c_i si la nueva $\mathrm{n}(c_i) \leq 1$. Otra forma de conseguirlo es excluir del proceso de agrupamiento aquellas palabras que ocurren un número pequeño de veces por considerarse poco fiables; además, eliminar las palabras menos frecuentes constituye un buen estimador de las palabras no vistas [KN93]. En nuestro caso, debido a que el proceso de agrupamiento se aplica sobre dominios restringidos, hemos preferido tener una mayor cobertura del lenguaje e

incorporar todas las palabras al proceso de agrupamiento garantizando de forma explicita que siempre se cumpla que $\mathrm{n}(c_i) > 1$.

Sustituyendo (3.21) y (3.22) en (3.18), la información mutua entre clases adyacentes cuando se aplica el método de dejar uno fuera, $\mathrm{I}_{lo}(c_1; c_2)$, queda:

$$\begin{aligned}\mathrm{I}_{lo}(c_1; c_2) \simeq \frac{1}{N-1} \Bigg[& \sum_{c_1c_2:\mathrm{n}(c_1c_2)>1} \mathrm{n}(c_1c_2) \log \frac{\mathrm{n}(c_1c_2) - 1 - b}{N-2} \\ & + n_1 \log \frac{(n_+ - 1)b}{(n_0+1)(N-2)} \\ & - 2 \sum_c \mathrm{n}(c) \log \frac{\mathrm{n}(c) - 1}{N-2} \Bigg]. \end{aligned} \tag{3.23}$$

El valor del parámetro b se puede estimar como [YB97, pág. 189]:

$$b = \frac{n_1}{n_1 + 2n_2}. \tag{3.24}$$

Aunque, a efectos prácticos, en lugar de entrenar b para cada agrupamiento, se suele determinar empíricamente un valor apropiado.

Hemos llamado al segundo de los algoritmos propuestos, algoritmo *iterativo dejando uno fuera* de agrupamiento monolingüe, precisamente por que hace uso del método de *dejar uno fuera* para calcular la función objetivo que guía el agrupamiento. De hecho, salvo por la nueva función objetivo, $\mathrm{I}_{lo}(c_1; c_2)$ frente a $\mathrm{I}(c_1; c_2)$, el algoritmo iterativo dejando uno fuera, es idéntico al algoritmo 3.1 descrito en la sección anterior. Así mismo, también utiliza las técnicas descritas en la sección anterior para la reducción del coste temporal puesto que el incremento de $\mathrm{I}_{lo}(c_1; c_2)$ también puede calcularse de forma parcial. La única complejidad añadida es la de tener que calcular la variación de los valores n_+, n_1 y n_0 con cada movimiento evaluado.

A modo de ejemplo de la utilización de este algoritmo, la figura 3.3 muestra las distintas perplejidades obtenidas sobre la parte castellana del corpus castellano–inglés EuTrans I (ver capítulo 6). En la figura se representan: la perplejidad del conjunto de test utilizando un modelo de bigramas obtenido sobre el corpus de entrenamiento; la evolución de la perplejidad del test utilizando modelos de bigramas basados en un número distinto de clases; la perplejidad del entrenamiento utilizando modelos de bigramas basados en un número distinto de clases; y la perplejidad del entrenamiento utilizando un modelo de bigramas. Si se observa la perplejidad medida sobre el test de los modelos de bigramas basados en clases, se puede apreciar como disminuye al principio conforme el número de clases aumenta para después comenzar a ascender. También se puede observar como la perplejidad medida sobre el conjunto de entrenamiento utilizando la técnica de dejar uno fuera presenta un comportamiento similar a la medida sobre el test. Además, si se compara la figura 3.3 con la la figura 3.2

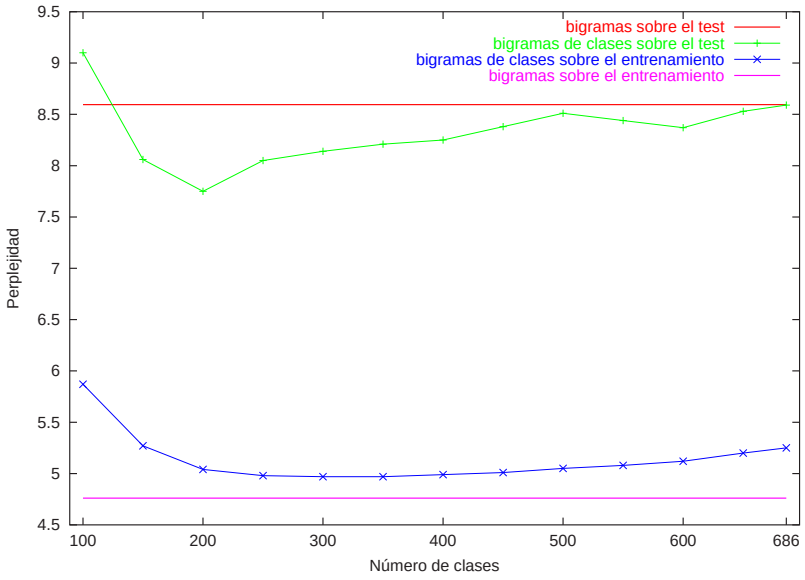


Figura 3.3: Perplejidad medida sobre el test y el entrenamiento para bigramas y bigramas de clases utilizando el algoritmo iterativo dejando uno fuera (comparar con la figura 3.2).

se puede ver como la perplejidad medida sobre el test es generalmente menor cuando se utiliza la técnica de dejar uno fuera. Por último, la perplejidad sobre el test alcanza su mínimo para 200 clases, valor relativamente cercano a las 300 clases en las que alcanza el mínimo la perplejidad sobre el corpus de entrenamiento. Por lo tanto, la perplejidad obtenida utilizando la técnica de dejar uno fuera permite obtener una estimación apropiada del número óptimo de clases.

3.6. Algoritmo incremental

El último de los algoritmos presentados, al que hemos bautizado con el nombre de algoritmo incremental (ver algoritmo 3.2), genera el agrupamiento de forma incremental; comienza con un número reducido de clases y crea nuevas clases, una a una, hasta alcanzar el número deseado de éstas.

Los dos algoritmos anteriores presentan una gran dependencia con la distribución inicial de palabras en clases. Esto es así ya que realizan una serie de movimientos de palabras, a partir de dicha distribución, hasta que se alcanza un máximo de la función objetivo; que puede ser un máximo local. Cuanto mejor sea la distribución inicial, más probabilidades habrá de que el agrupamiento alcanzado sea óptimo. A falta de más información, los dos algoritmos anteriores

optan por la utilización de una distribución inicial que permite que las palabras más frecuentes sean movidas en primer lugar. Asumimos que podemos sacar mejores conclusiones sobre el modelo del lenguaje a partir de aquellas palabras que hemos visto más veces.

Para contrarrestar la dependencia de estos algoritmos con la distribución inicial podríamos generar un número determinado de distribuciones iniciales al azar y realizar agrupamientos para cada una de estas distribuciones. De éstos, consideraríamos como el mejor agrupamiento a aquel que maximizara la función objetivo. Cuanto más grande sea el número de distribuciones iniciales, más probable será que encontremos un mejor agrupamiento. Sin embargo, como se ha visto en (3.2), el número de particiones posibles es extremadamente grande, por lo que esta aproximación no parece demasiado prometedora.

Frente a esta alternativa, planteamos un algoritmo que comienza con un número de clases menor que el que se pretende alcanzar y encuentra el agrupamiento (sub)óptimo para este número inicial de clases utilizando cualquiera de los dos algoritmos anteriores. Partiendo de dicho agrupamiento se crea una nueva clase a la que se mueven aquellas palabras que con su movimiento mejoren la función objetivo. De nuevo, se encuentra el agrupamiento (sub)óptimo para el nuevo número de clases utilizando cualquiera de los algoritmos anteriores. Así, hasta alcanzar el número de clases deseado. De esta forma, cada uno de los agrupamientos realizados comienza con una distribución inicial basada en el agrupamiento anterior y además, puede variar dicha distribución por completo si fuera conveniente.

Podríamos clasificar a este algoritmo como un algoritmo de agrupamiento jerárquico divisivo. Comienza con un determinado número de clases menor al deseado (en el extremo con sólo una clase) y termina cuando alcanza el número deseado de clases o cuando la creación de una nueva clase no mejora la función objetivo utilizada. Al igual que en el algoritmo jerárquico aglomerativo de Brown et al. [BDd$^+$92], y a diferencia de los agrupamientos jerárquicos clásicos, no estamos interesados en la obtención de una jerarquía de clases, únicamente en la distribución final de palabras en clases. Esta es la razón por la que podemos mover palabras de una clase a otra con cada incremento del número de clases.

El objetivo del algoritmo con cada incremento en el número de clases es encontrar una distribución con $n+1$ clases que mejore la distribución actual de n clases. Para ello, se añade una nueva clase c_{n+1} y se busca qué palabra w_i mejora la función objetivo al ser movida de su clase actual c_i a la nueva clase c_{n+1}. Una vez se ha movido la palabra w_i, se intentan mover las restantes palabras w_j de c_i a la clase c_{n+1}. De esta forma, se reparten los miembros de la clase c_i entre las clases c_i y c_{n+1}. Por último, sobre la nueva distribución de palabras en $n+1$ clases se realizan tantas iteraciones de cualquiera de los algoritmos descritos en las secciones anteriores como sean necesarias. Cuando no se realizan más movimientos o se alcanza el número de iteraciones prefijado, se incrementa en uno el número de clases y se repite todo el proceso.

Si se utiliza $\mathrm{I}(c_1; c_2)$ como función objetivo, en la forma descrita en la sec-

ción 3.4, tenemos el problema de que dicha función aumenta conforme aumenta el número de clases por lo que el número de clases deseado deberá ser fijado de antemano. Por otro lado, si se utiliza $\mathrm{I}_{lo}(c_1; c_2)$ como función objetivo, en la forma descrita en la sección 3.5, el algoritmo incremental puede parar de forma automática en el momento en el que $\mathrm{I}_{lo}(c_1; c_2)$ no mejore. En la experimentación que hemos realizado, hemos utilizado siempre este método conjuntamente con la función $\mathrm{I}_{lo}(c_1; c_2)$.

Algoritmo 3.2 Agrupamiento incremental

Requiere: C, $maxIter$
  Inicializar $\mathcal{G} : \mathcal{G}(w_i) \leftarrow 1,\ \forall w_i;\ n \leftarrow 1$
  repetir
    Crear una nueva clase c_{n+1} vacía
    Mover tentativamente cada palabra w a la nueva clase c_{n+1}
    Mover la palabra w_i que más incremente $\mathrm{I}_{lo}(c_1; c_2)$ a c_{n+1}
    si existe tal palabra w_i **entonces**
      para cada w_j en la anterior clase de w_i, c_i, **hacer**
        Mover w_j a c_{n+1} si con ello se incrementa $\mathrm{I}_{lo}(c_1; c_2)$
      fin para
      $n \leftarrow n + 1$
      Hacer *iterativo dejando uno fuera*$(n, maxIter)$
    fin
  hasta $n = C$ o $w_i = \emptyset$

3.7. Conclusiones

En este capítulo hemos presentado tres algoritmos de agrupamiento monolingüe. Los dos primeros de ellos difieren en la forma en la que calculan la función objetivo que utilizan. El algoritmo *iterativo* utiliza la información mutua entre clases adyacentes. El algoritmo *iterativo dejando uno fuera* utiliza dicha función objetivo calculada utilizando la técnica de dejar uno fuera para simular eventos no vistos y evitar el sobreentrenamiento. Se espera que el agrupamiento generado por el segundo de los métodos sea mejor que el generado por el primero. Además, también se espera que la forma en la que se calcula la función objetivo en el algoritmo *iterativo dejando uno fuera* permita estimar el número óptimo de clases a partir del corpus de entrenamiento.

El último de los algoritmos, el algoritmo *incremental* puede verse como una extensión de los anteriores que intenta minimizar el efecto de la distribución inicial sobre el agrupamiento obtenido. Comienza con un número de clases menor que el deseado e incrementa el número de clases de una en una hasta alcanzarlo. Se espera que los agrupamientos generados mediante el algoritmo *incremental* sean mejores que los obtenidos con cualquiera de los dos anteriores. Además, se

confía en que los agrupamientos generados por el algoritmo *incremental* para distintos números de clases sean más homogéneos que los obtenidos de forma independiente utilizando cualquiera de los otros dos métodos.

Capítulo 4

Agrupamiento bilingüe

4.1. Introducción

En el capítulo anterior se mostraron algoritmos capaces de agrupar automáticamente las palabras de un corpus monolingüe. El objetivo de dicho agrupamiento era la mejora de un modelo de lenguaje. En este capítulo se presenta una forma de aplicar dichos algoritmos para el caso de corpora bilingües. El objetivo de este agrupamiento será la mejora de un modelo de traducción. También se mostrará como integrar estas clases bilingües como parte de un modelo de traducción basado en traductores de estados finitos.

Mediante el agrupamiento bilingüe se pretende encontrar un conjunto de clases tanto para el idioma origen como el destino de modo que dicha agrupación sirva para la obtención de mejores traductores.

Para que las clases obtenidas sean útiles desde el punto de vista de la traducción es necesario que el método de agrupamiento tenga en cuenta las relaciones existentes entre los dos idiomas. No es suficiente con realizar un agrupamiento monolingüe en cada idioma por separado ya que los agrupamientos obtenidos en un idioma pueden no tener relación con los agrupamientos en el otro idioma. Para que exista esta relación, se determinan habitualmente los agrupamientos en uno de los dos idiomas en función de las categorías gramaticales predefinidas o agrupamientos automáticos ya obtenidos en el otro idioma.

El resto del capítulo está organizado como sigue. En la sección 4.2 se muestran algunos métodos de agrupamiento bilingüe. La sección 4.3 está dedicada a la descripción del método que nosotros proponemos, *e-cluster*. En la sección 4.4 se plantea la integración de dicho método con el modelo de traductores subsecuenciales. Finalmente, la última sección describe el proceso requerido para la obtención de las traducciones.

4.2. Clases y traducción

Puesto que el agrupamiento de las palabras de un idioma se ha utilizado con éxito en la obtención de mejores modelos de lenguaje, es lícito pensar que el agrupamiento de las palabras de dos lenguajes para los que se desea desarrollar un traductor, conducirá a la mejora de los modelos de traducción.

Sin embargo, aunque esta última suposición ha sido explotada con éxito, la aplicación de dicha idea no es tan inmediata como la de simplemente encontrar clases de forma independiente en ambos lenguajes. La correspondencia entre las clases del lenguaje origen y las del destino obtenidas de forma independiente no tiene por qué tener sentido para un modelo de traducción.

En el escenario más habitual de mejora de la traducción gracias a categorías, los textos de ambos lenguajes se etiquetan con sendos etiquetadores de categorías gramaticales. A partir de este texto etiquetado se obtiene una relación entre etiquetas de ambos lenguajes. Sin embargo, en el caso de categorías gramaticales, éstas son determinadas por humanos de acuerdo al conocimiento lingüístico que se posea de un determinado lenguaje. No resulta evidente que tenga que haber una correspondencia entre categorías gramaticales de dos idiomas, especialmente si la diferencia entre ambos es grande [FW90]. La relación derivada de las categorías gramaticales no tiene por qué suponer una buena restricción para la búsqueda de traducciones. Fung y Wu proponen en [FW90] un modelo al que denominan Modelo de Markov Coaccionado (CMM — *Coerced Markov Model*) que obtiene la relación existente entre palabras en el idioma origen (chino) y categorías gramaticales en el idioma destino (inglés). Dicho modelo predice, dada una frase en el idioma origen, la secuencia de categorías gramaticales correspondiente en el idioma destino. Esta secuencia se utiliza posteriormente para podar la búsqueda de una traducción de la frase origen.

En el caso de utilizar métodos de agrupamiento monolingües sobre cada uno de los idiomas de forma separada, los agrupamientos obtenidos reflejarán las características propias a cada uno de los lenguajes [WLW96] dificultando la posterior relación entre clases de ambos lenguajes. Utilizando restricciones de ambos lenguajes para la búsqueda de los agrupamientos se obtienen clases de mejor calidad al eliminar aquellos usos específicos que se den en uno sólo de los lenguajes. Además, las clases resultantes seguramente serán más apropiadas para la traducción automática.

Wang, Lafferty y Waibel proponen en [WLW96] un método de agrupamiento bilingüe basado en la optimización de la información mutua entre clases adyacentes. Utilizan el alineamiento entre las frases de ambos idiomas como nexo de unión entre las clases de un idioma y las palabras del otro. En lugar de maximizar el logaritmo de la verosimilitud de uno de los lenguajes dada una función de agrupamiento —el método utilizado en el agrupamiento monolingüe— proponen utilizar el logaritmo de la verosimilitud conjunta de ambos lenguajes dada la función de agrupamiento sobre uno de ellos. Puesto que con cada cambio en la función de agrupamiento es necesario reestimar los parámetros del modelo

de traducción, se ven obligados a realizar una serie de suposiciones para no recalcular dichos parámetros con cada movimiento: lo que dispararía las exigencias computacionales.

El método propuesto en [WLW96] agrupa palabras de un idioma utilizando los alineamientos entre los grupos de ese idioma y las palabras del otro. Para obtener un agrupamiento en el otro idioma, se repite el mismo planteamiento pero a la inversa. Por lo tanto, las clases obtenidas en ambos idiomas están influenciadas por los alineamientos entre las palabras de ambos idiomas; sin embargo, no se obtienen conjuntamente.

Och y Weber [OW98] también consideran que si se desea utilizar agrupamientos de palabras en traducción parece tener sentido que los agrupamientos de palabras en ambos idiomas estén fuertemente correlacionados; de esta forma, la información sobre la clase a la que pertenece una determinada palabra del idioma fuente proporcionaría mucha información sobre la clase a la que pertenece la palabra del idioma destino que se debe generar.

Siguiendo su argumentación, las clases obtenidas de forma independiente para los lenguajes fuente y destino, basadas en las cuentas de palabras adyacentes $\mathrm{n}(w_{i-1}, w_i)$, presentan la propiedad de que el saber que una determinada palabra w_{i-1} pertenece a una clase $c_{w_{i-1}}$ proporciona una gran información sobre la clase a la que pertenecerá la siguiente palabra w_i. Sin embargo, las clases en ambos idiomas no están correlacionadas. Para obtener clases en ambos idiomas que estén correlacionadas, definen sendas nuevas funciones de cuentas, $N_{x \to y}(x, y) = \mathrm{P}(y \mid x) \cdot \mathrm{n}(y)$ y $N_{y \to x}(x, y) = \mathrm{P}(x \mid y) \cdot \mathrm{n}(x)$, con las que obtienen clases que están fuertemente correlacionadas. Sin embargo, raramente contienen palabras con propiedades sintácticas/semánticas similares.

Para conseguir clases de palabras que presenten ambas propiedades, Och y Weber [OW98] proponen utilizar el primer método sobre el lenguaje destino y una vez obtenidas dichas clases, utilizar el segundo método para obtener las clases en el lenguaje origen.

En [Och99], partiendo de la suposición de que existe una función de alineamiento entre palabras conocida, a_1^J, y que cada palabra del idioma destino, y_j, es generada por la palabra x_{a_j} del idioma fuente con una determinada probabilidad $\mathrm{P}(y_j \mid x_{a_j})$, se llega a una función de optimización sobre un conjunto de clases que engloba tanto a las clases del idioma destino como del idioma fuente —durante el proceso de optimización se impide que las palabras de un idioma se muevan a las clases del otro—. El algoritmo empleado es similar al algoritmo *iterativo* descrito en el capítulo anterior con la salvedad de que cada iteración se realiza primero sobre todas las palabras del idioma destino y luego sobre todas las palabras del idioma fuente. Dicho planteamiento es una extensión y mejora de [OW98]; y se propone como una alternativa más simple y eficiente desde el punto de vista computacional que [WLW96].

También se propone en [Och99] una aproximación en dos pasos: en el primero se determina un agrupamiento óptimo de las clases del idioma destino utilizando la función de optimización descrita para el caso monolingüe y en el segundo se

utiliza dicho agrupamiento óptimo y la función de optimización bilingüe para obtener el agrupamiento en el idioma fuente. Utilizando esta aproximación en dos pasos se fuerza a que las clases del idioma destino sean «buenas» desde el punto de vista monolingüe y que las clases del idioma origen estén fuertemente correladas con las del idioma destino. Es interesante destacar que el uso de esta aproximación en los experimentos realizados en [Och99] conlleva traducciones de mejor calidad que los obtenidos mediante el agrupamiento simultáneo de las palabras en ambos idiomas.

El método que se propone en esta tesis, al que hemos bautizado *e-cluster*, y que se describe en la siguiente sección, se basa en la suposición de que existe una función de alineamiento entre palabras, que puede determinarse de forma automática, y pretende la obtención de agrupamientos idóneos en el corpus destino y agrupamientos correlacionados con estos en el idioma origen.

4.3. El método *e-cluster*

El método propuesto en esta tesis, *e-cluster*, parte de la suposición de que existe una función de alineamiento $\vec{a}$ que relaciona cada palabra destino y_j con una palabra fuente x_{a_j}. En base a dichos alineamientos, que como se ha visto pueden determinarse de forma automática, la idea es etiquetar las palabras del corpus destino con las palabras del corpus origen con la que han sido alineadas. Dichas palabras etiquetadas, a las que llamamos palabras extendidas (*e-words*), presentan la peculiaridad de que matizan el significado y/o función de la palabra original mediante su traducción en otro idioma. Por ejemplo, al traducir de inglés a español, podrían aparecer las palabras extendidas $\textit{banco}_{\textit{bank}}$ y $\textit{banco}_{\textit{bench}}$. De esta forma, el significado de la palabra original (*banco*), queda definido claramente. Al nuevo corpus formado por las palabras extendidas le hemos llamado corpus extendido (*e-corpus*).

Se pueden obtener clases de palabras extendidas simplemente utilizando un agrupamiento monolingüe sobre el corpus extendido. En el método propuesto, cada palabra extendida es asignada a una clase, pero una palabra del idioma origen o del idioma destino puede pertenecer a más de una clase si aparece formando parte de más de una palabra extendida. Esto sucederá en aquellos casos en los que una palabra tiene varias traducciones. De esta forma, frente a los métodos descritos en la sección anterior en la que cada palabra era asignada a una única clase, el método propuesto permite diferenciar distintas acepciones o comportamientos de una misma palabra en función de su traducción.

Estas clases de palabras extendidas pueden verse como agrupamientos de pares de palabras; donde una de las palabras pertenece al idioma destino y la otra es una de sus posibles traducciones en el idioma origen. Si nos fijamos en las palabras correspondientes al idioma destino, éstas han sido agrupadas de acuerdo a dicho lenguaje, pudiendo una palabra, aparecer en más de un grupo, si en el otro idioma se traduce de forma distinta dependiendo de su función. Si,

por el contrario, nos fijamos en las palabras del idioma origen, éstas han sido agrupadas de acuerdo a los grupos obtenidos para el idioma destino por lo que se obtiene la mayor correlación posible: la clase de una palabra origen es la clase de su traducción en el lenguaje destino.

La propiedad que se desea que presenten dichas agrupaciones y que permita compensar la escasez de muestras de entrenamiento inherentes al proceso de traducción es la siguiente: si en una determinada frase hay un par (x_i, y_i) perteneciente a clase c, la traducción de la frase origen sustituyendo la palabra x_i por otra x_j de la misma clase se obtiene sin más que reemplazar y_i por y_j en la traducción original. Por ejemplo, si se sabe que la traducción de la frase *el* bolígrafo *azul* es *the blue* pen y el par *(bolígrafo, pen)* pertenece a la misma clase que *(lápiz, pencil)*, entonces la traducción de *el lápiz azul* debería obtenerse sin más que cambiar *pen* por la pareja de *lápiz*, quedando por tanto, *the blue pencil*.

El método propuesto *e-cluster* para la obtención de agrupamientos bilingües consta de las siguientes fases (ver figura 4.1):

- Alineamiento del corpus bilingüe.
- Generación de un corpus extendido.
- Agrupamiento de las palabras extendidas.

El alineamiento del corpus de entrenamiento consiste en la determinación por medio de métodos estadísticos de la probabilidad de que una determinada palabra del idioma destino sea la traducción de una determinada palabra del idioma origen.

Utilizando dicho alineamiento se puede generar un nuevo corpus formado por las palabras de salida etiquetadas con aquellas palabra de entrada con las que han sido alineadas.

Sobre este corpus extendido se puede realizar un agrupamiento monolingüe utilizando alguno de los algoritmos descritos en el capítulo anterior. Conviene tener en cuenta, sin embargo, que el corpus extendido no estará formado en su totalidad por palabras extendidas ya que aquellas palabras que no hayan sido alineadas con ninguna aparecerán tal cual en el corpus extendido. En el algoritmo de agrupamiento se deberá establecer la estrategia que defina cómo intervienen dichas palabras en el proceso de agrupamiento.

4.3.1. Alineamiento del corpus bilingüe

Para cada palabra del idioma destino es posible utilizar métodos estadísticos para determinar de cuál de las palabras del idioma origen es traducción. Dicha relación recibe el nombre de alineamiento.

Estos alineamientos pueden obtenerse utilizando los distintos modelos de traducción propuestos por Brown et al. [BDDM93]. Para ello, sólo es necesario disponer de un corpus bilingüe en el que para cada frase del idioma origen se conozca su traducción en el idioma destino.

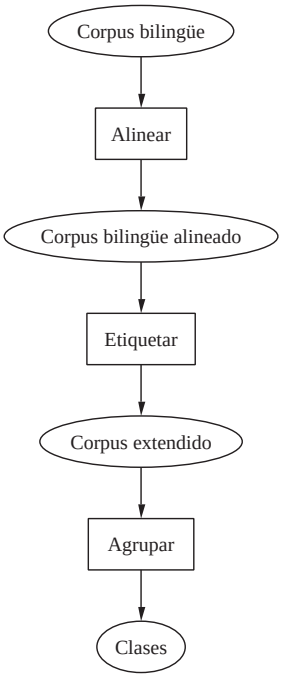


Figura 4.1: Esquema del método *e-cluster* de agrupamiento bilingüe.

Dependiendo del sentido en el que se realice el alineamiento, distinguiremos tres tipos de alineamientos:

Directo Las palabras de la frase de salida se alinean con las de la frase de entrada. Cada palabra de la frase de salida está alineada con una única palabra de entrada; sin embargo, varias palabras de la frase de salida pueden estar alineadas con la misma palabra de la frase de entrada.

Inverso El contrario del anterior: se alinean las palabras de la frase de entrada con las de salida.

Cruzado Es el alineamiento que se obtiene al cruzar los alineamientos *directo* e *inverso* dejando únicamente aquellos alineamientos que se dan en los dos sentidos.

Para obtener el alineamiento inverso basta con invertir el orden con el que se presentan los dos idiomas al alineador.

El alineamiento cruzado supone un alineamiento entre palabras una a una, mientras que tanto el alineamiento directo como el inverso permiten que varias

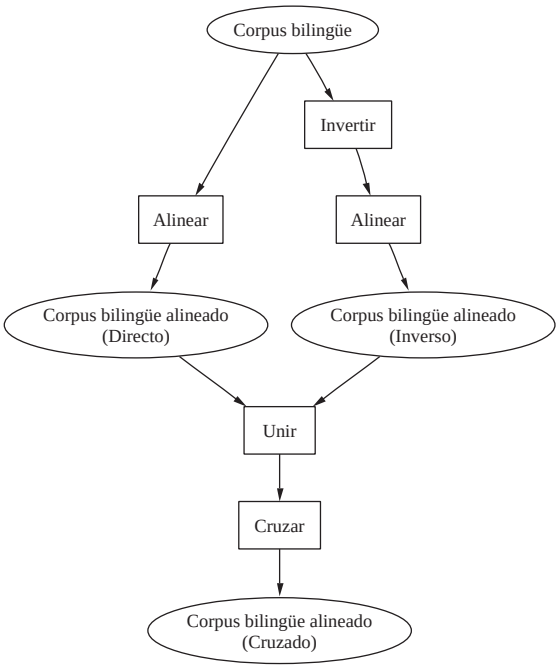


Figura 4.2: Esquema del método de obtención de los alineamientos.

palabras estén alineadas con una misma palabra. Dado un alineamiento directo, $\vec{d}$, y otro inverso, $\vec{i}$, definimos el alineamiento cruzado, $\vec{c}$, como:

$$\vec{c} = \left\{ \vec{c}_j : 1 \leq j \leq |\vec{d}| \right\}, \tag{4.1}$$

donde $\vec{c}_j$ se define como:

$$\vec{c}_j = \begin{cases} \vec{d}_j & \text{si } \vec{i}_{\vec{d}_j} = j \\ 0 & \text{en otro caso.} \end{cases} \tag{4.2}$$

El alineamiento cruzado proporciona generalmente menos alineamientos que el directo. Sin embargo, los alineamientos del cruzado se consideran más fiables; han sido encontrados en ambos sentidos.

La figura 4.2 muestra el esquema general para la obtención de los tres tipos de alineamientos mencionados. A modo de ejemplo, la figura 4.3 muestra los alineamientos directo, inverso y cruzado obtenidos sobre un par de frases del corpus EuTrans I (descrito en el capítulo 6) utilizando el modelo 2 de IBM.

El alineamiento producido por el modelo 2 de IBM se puede mejorar utilizando un criterio de vecindad que revise aquellos alineamientos que puedan ser

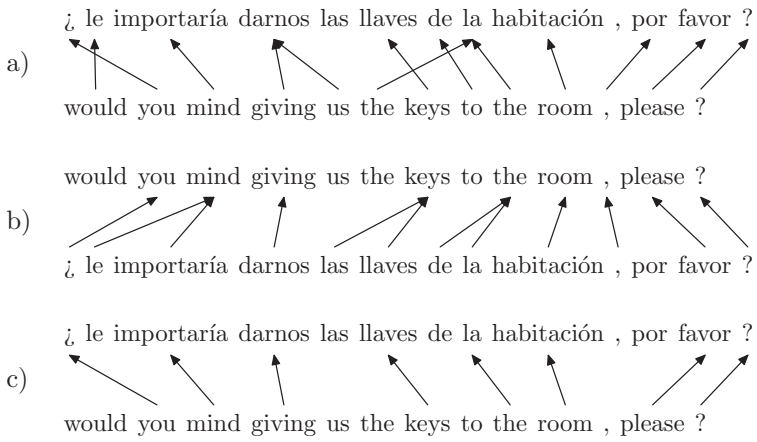


Figura 4.3: Ejemplo de frases alineadas con el modelo 2 de IBM: a) directo, b) inverso y c) cruzado.

erróneos. Para ello, proponemos un algoritmo, al que hemos llamado *neighbours* (ver algoritmo 4.1). La idea de este algoritmo es la de explicar mejor aquellas palabras que han sido alineadas o bien con la palabra vacía o bien con una misma palabra. Este algoritmo utiliza un diccionario y los alineamientos proporcionados por el modelo 2 de IBM para buscar un mejor alineamiento para estas palabras.

La idea principal es que la palabra con la que debe alinearse la palabra y_i es probable que esté en las proximidades de las palabras con las que se alinean sus vecinas. El algoritmo supone que el alineamiento proporcionado por el modelo 2 es cercano al ideal y es por eso que sólo intenta encontrar nuevos alineamientos para aquellas palabras que no han sido alineadas y para aquellas en las que varias están alineadas con la misma palabra. Hay que tener en cuenta que este último caso puede ser perfectamente válido: varias palabras en un idioma pueden ser debidas a una única palabra en otro idioma; sin embargo, las deficiencias que se pretenden paliar son aquellas que se dan cuando una misma palabra y su traducción aparecen varias veces en una frase, y todas las traducciones se alinean con la misma aparición de la palabra que ha sido traducida.

El algoritmo 4.1 requiere las frases de entrada y salida, x e y, el alineamiento directo $\vec{a}_{y \to x}$, y la distribución de probabilidad $\mathrm{P}(y \mid x)$. La salida del algoritmo es el nuevo alineamiento para el par de frases de entrada. A modo de ejemplo, la figura 4.4 muestra los alineamientos directo, inverso y cruzado obtenidos sobre un par de frases del corpus EuTrans I utilizando el modelo 2 de IBM y el algoritmo *neighbours*.

Si comparamos los alineamientos generados por el modelo 2 de IBM (figura 4.3) con los alineamientos obtenidos si sobre dicho modelo se aplica el algoritmo *neighbours* (figura 4.4), se puede observar como el alineamiento en general

Algoritmo 4.1 Alineamiento por vecindad —*neighbours*—

Requiere: x, y, $\vec{a}_{y\to x}$ y P
  $\vec{b}_{y\to x} \leftarrow \vec{a}_{y\to x}$
  para cada $i : \vec{a}_{y\to x}[i] = 0 \vee \exists j \mid \vec{a}_{y\to x}[i] = \vec{a}_{y\to x}[j]$ **hacer**
    $j' \leftarrow \begin{cases} \text{máx}\{j : 1 \leq j < i \wedge \vec{a}_{y\to x}[j] \neq 0\} & \text{si } i > 1 \\ 1 & \text{en otro caso} \end{cases}$
    $lim_1 \leftarrow \vec{a}_{y\to x}[j']$
    $j' \leftarrow \begin{cases} \text{mín}\{j : i < j \leq \mid y \mid \wedge \vec{a}_{y\to x}[j] \neq 0\} & \text{si } i < \mid y \mid \\ \mid y \mid & \text{en otro caso} \end{cases}$
    $lim_2 \leftarrow \vec{a}_{y\to x}[j']$
    $lim_{izq} = \text{mín}(lim_1, lim_2)$
    $lim_{der} = \text{máx}(lim_1, lim_2)$
    $\vec{b}_{y\to x}[i] \leftarrow \arg\max_{j\in[lim_{izq}, lim_{der}]} P(y_i \mid x_j)$
  fin para
  devuelve $\vec{b}_{y\to x}$

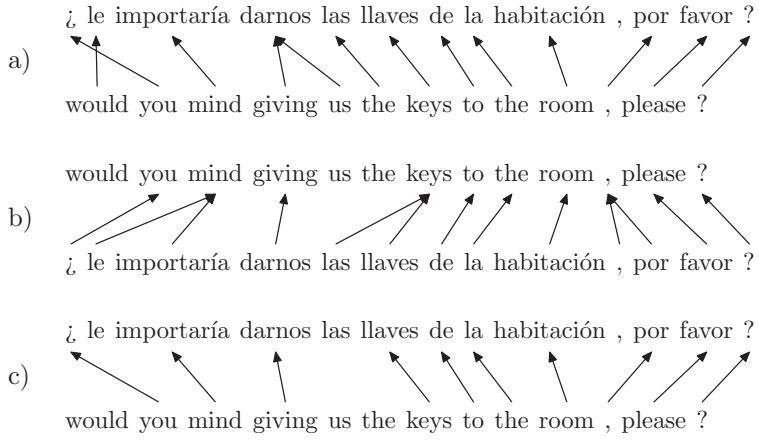


Figura 4.4: Ejemplo de frases alineadas con el modelo 2 de IBM y *neighbours*: a) directo, b) inverso y c) cruzado. Compárese con la figura 4.3.

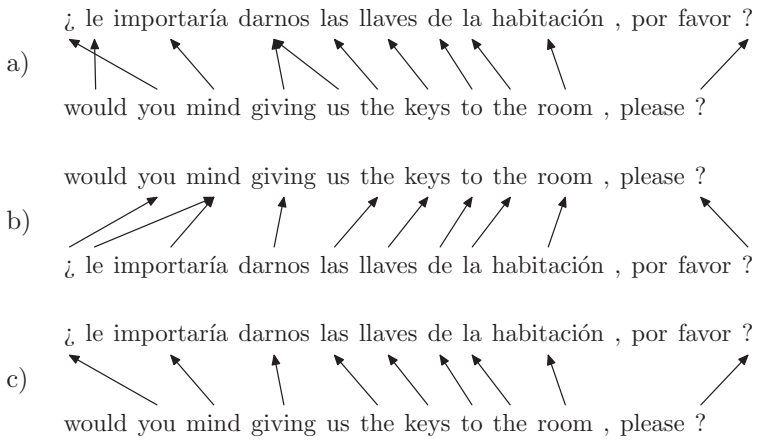


Figura 4.5: Ejemplo de frases alineadas con el modelo 3 de IBM: a) directo, b) inverso y c) cruzado.

ha mejorado. Por ejemplo, en el alineamiento directo, la palabra *the* que estaba incorrectamente alineada con la palabra *la*, pasa a estarlo con la correcta *las*. En el alineamiento inverso, las palabras *de* y *la* que estaban alineadas con *the* pasan a alinearse con *to* y *the*, respectivamente. Sin embargo, la palabra *por*, que no estaba alineada con ninguna, pasa a alinearse incorrectamente con el signo de puntuación «,». Esto es debido a que el diccionario generado de forma automática permite dicho alineamiento. Una vez aplicado el criterio de vecindad sobre los alineamientos directos e inverso, el alineamiento cruzado presenta el nuevo alineamiento correcto de la palabra *to* con *de* junto con el incorrecto del signo de puntuación «,» con la palabra *por*.

Para la realización de los experimentos mostrados en el capítulo 6 se han utilizado además de los dos métodos arriba indicados, los alineamientos producidos por los modelos más avanzados 3 y 4 de IBM. En lugar de utilizar únicamente el modelo 4 de IBM se ha preferido realizar experimentos también con los otros modelos de alineamiento. De esta forma, se puede observar el efecto de éstos sobre las traducciones obtenidas y se pueden comparar los resultados presentados en esta tesis con los publicados previamente; estos últimos utilizaban los modelos 2 y 2 con *neighbours*.

A modo de ejemplo, las figuras 4.5 y 4.6 muestran los alineamientos directo, inverso y cruzado obtenidos sobre un par de frases del corpus EuTrans I utilizando los modelo 3 y 4 de IBM, respectivamente.

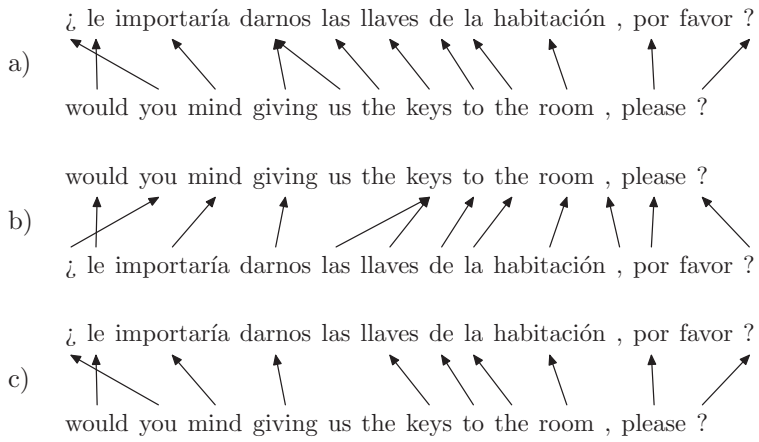


Figura 4.6: Ejemplo de frases alineadas con el modelo 4 de IBM: a) directo, b) inverso y c) cruzado.

$$would\ you_{\,?}\ mind_{\,importaría}\ giving_{\,darnos}\ us\ the_{\,las}\ keys_{\,llaves}\ to_{\,de}\ the_{\,la}\ room_{\,habitación}\ ,\ please\ ?_{\,?}$$

Cuadro 4.1: Frase del corpus extendido debida al alineamiento cruzado mostrado en la figura 4.5

4.3.2. Generación de un corpus extendido

Para la generación del corpus extendido se utiliza el alineamiento cruzado comentado en la sección anterior. Las palabras del idioma destino son etiquetadas con aquellas palabras del idioma origen con las que han sido alineadas. El corpus resultante estará formado por palabras del idioma destino no alineadas y por palabras del idioma destino etiquetadas con las palabras del idioma origen con las que habían sido alineadas. A éstas últimas, las hemos denominado *palabras extendidas* ya que, debido a que están etiquetadas con la palabra con la que han sido alineadas, proporcionan mayor información que por sí solas. De esta forma, si viéramos la palabra extendida $light_{\,luz}$, sabríamos que nos estamos refiriendo a la palabra *light* cuando aparece como traducción de *luz*. Es decir, el etiquetado permite diferenciar varios usos de una misma palabra. Siempre y cuando en el idioma origen éstos se representan por distintas palabras. Al corpus así generado lo hemos denominado *corpus extendido*.

A modo de ejemplo, el cuadro 4.1 muestra como quedaría en el corpus extendido el par de frases alineadas por el modelo 3 de IBM mostradas en la figura 4.5c. Como se puede observar, la palabra *the* aparece ahora como dos palabras extendidas diferentes: $the_{\,las}$ y $the_{\,la}$, dependiendo de si la palabra *the* aparece como traducción de *las* o de *la*.

1 :	would $_{\text{le}}$ would $_{\text{les}}$ can $_{\text{podría}}$ could $_{\text{podrían}}$ can $_{\text{puede}}$ can $_{\text{pueden}}$ will $_{\text{quiere}}$ will $_{\text{quieren}}$ do $_{\text{tiene}}$
50 :	included $_{\text{anotado}}$ included $_{\text{apuntado}}$
100 :	give $_{\text{darme}}$ wake $_{\text{despertarme}}$
150 :	make $_{\text{hágame}}$ make $_{\text{háganos}}$
200 :	leave $_{\text{irnos}}$ leave $_{\text{marcharnos}}$
250 :	not $_{\text{nada}}$

Cuadro 4.2: Ejemplos de clases de palabras extendidas.

4.3.3. Agrupamiento de las palabras extendidas

Una vez se dispone de un corpus extendido es fácil realizar un agrupamiento de las palabras extendidas. Para ello, es suficiente con aplicar cualquiera de los métodos de agrupamiento monolingüe vistos en el capítulo anterior sobre el corpus extendido.

El único detalle que conviene tener en cuenta es la existencia de palabras en el corpus extendido que no son palabras extendidas. Puesto que lo que queremos es obtener un agrupamiento sólo de las palabras extendidas, se ha optado por asignar a las palabras no extendidas cada una a una clase propia. Durante el agrupamiento dichas palabras permanecerán siempre en su clase. De esta forma, dichas palabras intervienen en el cálculo de la función de agrupamiento pero no son movidas de unas clases a otras.

Una vez realizado el agrupamiento nos quedaremos únicamente con aquellas clases que estén formadas por palabras extendidas. De esta forma, habremos realizado un agrupamiento que involucra simultáneamente a palabras del idioma destino y a las palabras del idioma origen de las que son traducción.

Para ver un ejemplo del tipo de agrupamiento que se obtiene, hemos alineado con el modelo 4 de IBM el corpus EuTrans I (ver capítulo 6), hemos generado el corpus extendido y hemos utilizado el algoritmo de agrupamiento *incremental dejando uno fuera* comentado en el capítulo anterior con las modificaciones comentadas en esta sección. El número de clases óptimo se ha obtenido de forma automática. El cuadro 4.2 muestra algunas de las clases obtenidas entresacadas al azar de las 269 generadas.

En la clase 1 vemos como se han agrupado en la misma clase los auxiliares *could*, *can*, *do*, *will* y *would* cuando han sido alineados con: *podrían* en el caso de *could*; *podría*, *puede* y *pueden* en el caso de *can*; *tiene* en el caso de *do*; *quiere* y *quieren* en el caso de *will*; y *le* y *les* en el caso de *would*. Estas palabras extendidas se han generado a partir de frases interrogativas del tipo *nos podrían . . .* traducida por *could you . . .* , etc.

La clase 50 contiene al participio *included* cuando ha sido alineado con *anotado* o *apuntado*. Estas palabras extendidas provienen de pares de frases del tipo:

¿está todo anotado? traducida por *is everything included?*.

La clase 150 contiene al verbo *make* cuando es traducción de *hágame* o de *háganos*. Por ejemplo, en pares de frases del tipo: *por favor, hágame la cuenta de la habitación setecientos veinte.* traducida como *could you make out the bill for room number seven two oh for me, please?*.

La clase 200 contiene a las palabras extendidas $leave_{irnos}$ y $leave_{marcharnos}$ correspondientes a la tercera persona del plural de los verbos ir y marchar, respectivamente.

La clase 250 sólo tiene a la palabra extendida not_{nada} proveniente de pares de frases del tipo: *de nada* traducida por *not at all*.

El agrupamiento completo puede observarse en la sección A.2. Algunos de los agrupamientos realizados son:

- Los días de la semana están todos en la clase 37.
- Los meses del año están todos en la clase 235.
- Los nombres propios de personas que aparecían en el corpus se han repartido entre las clases 7, 261 y 267.
- Los apellidos, entre las clases 12 y 262.
- Las características relacionadas con el contenido de habitaciones están en la clase 16: $bath_{aseo}$, $minibar_{bar}$, $bath_{baño}$ ...
- Los adjetivos referidos a habitaciones en la clase 116: $hot_{calurosa}$, $expensive_{cara}$, $expensive_{caras}$ ...

Además, se pueden observar varios ejemplos en los que determinadas palabras aparecen en más de una clase dependiendo de si forman parte de más de una palabra extendida. Así por ejemplo, las palabras inglesas *double* y *single* aparecen en las clases 33 y 40 dependiendo de si son traducción en castellano de las palabras en singular: doble e individual o de las palabras en plural: dobles e individuales.

4.4. Inferencia de un SST con clases

En esta sección veremos como se puede inferir un transductor subsecuencial que haga uso de las clases de palabras extendidas obtenidas en la sección anterior. El procedimiento a seguir se muestra en la figura 4.7. En primer lugar, se genera un corpus bilingüe etiquetado con las clases correspondientes. A continuación, se infiere un autómata sobre dicho corpus. Después, para cada una de las clases se crea un autómata trivial. Y por último, se expande el autómata inferido con los autómatas triviales.

El transductor subsecuencial así generado, aceptará frases del idioma origen y devolverá su correspondiente traducción.

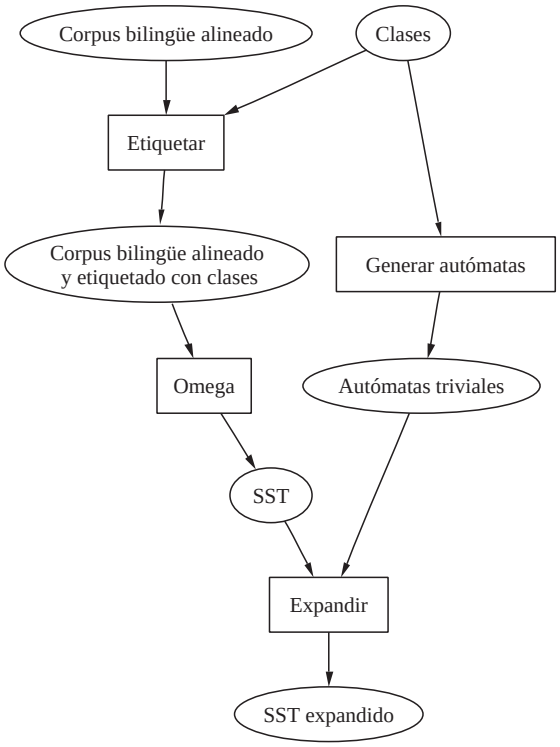


Figura 4.7: Esquema de la obtención del traductor utilizando las clases generadas por *e-cluster*.

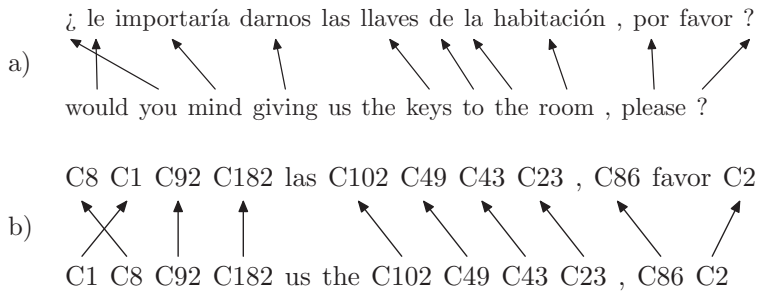


Figura 4.8: Ejemplo de sustitución de pares de palabras por sus correspondientes etiquetas: a) frase alineada cruzada (figura 4.6c) y b) frase etiquetada con clases.

4.4.1. Generación del corpus bilingüe etiquetado con clases

El primero de los pasos para la inferencia de un SST que utilice las clases de palabras extendidas obtenidas por *e-cluster* es la sustitución en el corpus bilingüe original de aquellas palabras que forman parte de una palabra extendida por una etiqueta de la clase a la que dicha palabra extendida pertenece.

Como se vio en la sección anterior, las palabras extendidas se extraen de un corpus bilingüe sobre el que se ha realizado un alineamiento cruzado. Por lo tanto, para etiquetar cada palabra extendida por su clase, bastará recorrer el corpus alineado cruzado y sustituir cada par de palabra origen, palabra destino, por una etiqueta de la clase a la que pertenecen. Por ejemplo, si partimos del par de frases alineadas por el método cruzado que se muestra en la figura 4.8a (alineamiento cruzado IBMm4) llegaremos al par de frases etiquetadas con clases que se muestra en la figura 4.8b.

Puesto que el alineamiento directo proporciona más información sobre alineamientos que el alineamiento cruzado, es interesante restaurar dicha información para la inferencia del SST. Siguiendo con el procedimiento descrito hasta ahora para el etiquetado del corpus con la información del agrupamiento, bastaría con copiar el alineamiento directo del corpus bilingüe original sobre el nuevo corpus etiquetado. Así, la frase representada en la figura 4.9b correspondería a la frase etiquetada con las clases obtenida en el paso anterior sobre la que se han añadido los alineamientos que se perdieron al realizar el alineamiento cruzado (la figura 4.9a muestra la frase original con el alineamiento directo).

El procedimiento descrito para el etiquetado del corpus es el que hemos utilizado hasta ahora [BV99a, BV99b, BV99c, BV01]. Se propone un par de mejoras sobre dicho procedimiento teniendo en cuenta las siguientes consideraciones:

- Es posible que distintos pares de frases generen en algún caso distintas palabras extendidas cuando *deberían* haber generado la misma. Imaginemos, por ejemplo, que *darnos* ha sido alineado de forma cruzada en algunas

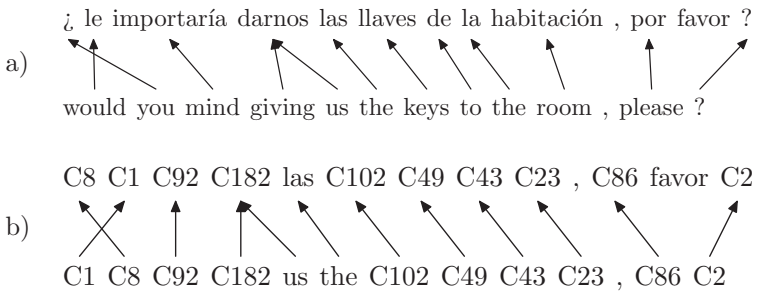


Figura 4.9: Ejemplo de recuperación del alineamiento directo sobre el corpus etiquetado con clases: a) frase alineada directa (figura 4.6a) y b) frase etiquetada con clases y alineamiento directo.

ocasiones con *giving* y otras con *us*. Si esto fuera así, y siguiendo el procedimiento descrito hasta ahora, cuando etiquetáramos las frases en el lugar de la palabra *darnos* colocaríamos una clase u otra dependiendo del alineamiento de esa frase en cuestión y dichas frases se convertirían en ejemplos de traducción distintos.

- Debido a que una palabra origen puede pertenecer a más de una clase sin más que pertenecer a más de una palabra extendida, puede ocurrir que una determinada palabra origen sea etiquetada con distintas clases. Esto es correcto si las clases están formadas por más de una palabra extendida. En caso contrario, el etiquetar dicha palabra origen con distintas clases dificulta la tarea del traductor ya que se entrena con distintas palabras origen cuando en realidad se trata de la misma.

Para tener en cuenta la primera de las consideraciones, el procedimiento que se sigue es el siguiente. Se parte del corpus bilingüe alineado por el método directo, no por el cruzado como se ha descrito anteriormente. Y en el caso de que más de una palabra del idioma destino esté alineada con una palabra del idioma origen sólo se sustituye por su clase correspondiente aquel par cuya palabra extendida se haya visto un mayor número de veces en el corpus. Así, por ejemplo, si en la frase 4.9a, $giving_{darnos}$ y us_{darnos} fueran palabras extendidas válidas, sólo se sustituiría aquel par de palabras que correspondieran a la palabra extendida más vista (si us_{darnos} ha sido poco vista, acabaríamos sustituyendo siempre *darnos* y *giving* por su clase correspondiente).

La segunda de las consideraciones expuestas, la de que una palabra origen no agrupada con otras sea etiquetada con distintas clases se resuelve, una vez seleccionadas las mejores palabras extendidas para cada frase, etiquetando sólo aquellas que posean más de un elemento. La figura 4.10b muestra la frase etiquetada siguiendo los pasos anteriores a partir de la frase mostrada en la figura 4.10a.

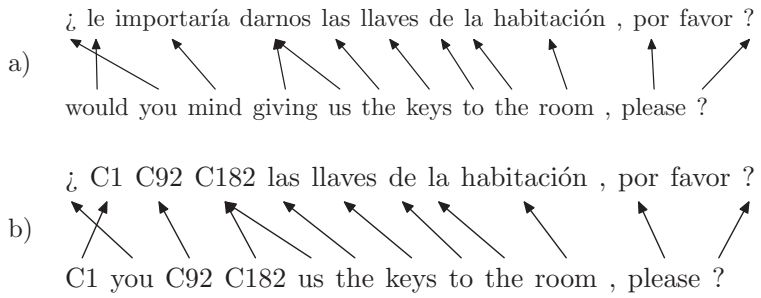


Figura 4.10: Ejemplo de recuperación del alineamiento directo sobre el corpus etiquetado con clases (método mejorado): a) frase alineada directa (figura 4.6a) y b) frase etiquetada con clases y alineamiento directo.

4.4.2. Inferencia del SST

Utilizando el corpus alineado etiquetado con clases obtenido en el paso anterior se puede inferir un transductor subsecuencial utilizando cualquiera de los algoritmos existentes para ello. Dichos algoritmos considerarán a las etiquetas de clases simplemente como palabras y generarán el correspondiente SST.

El algoritmo que se ha utilizado para el desarrollo de esta tesis ha sido el algoritmo OMEGA [Vil98, Vil00]. Este algoritmo puede verse como una mejora del algoritmo OSTIA [OGV93] y de ahí su nombre *OSTIA Modified for Employing Guarantees and Alignments*. Esta familia de algoritmos tienen por objetivo entrenar modelos estructurales adecuados partiendo de pares de frases origen/destino de entrenamiento. Los modelos entrenados por OMEGA son SSTs.

Se distinguen dos fases en el entrenamiento de dichos modelos:

1. Construcción de un árbol inicial. Una representación compacta de los pares de entrenamiento.

2. Unión de estados. Se realiza una serie de uniones de estados para generalizar el corpus de entrenamiento.

La idea básica para el guiado de la unión de estados es la de tener en cuenta los modelos de lenguaje de los idiomas origen y destino. El algoritmo recorre el árbol de estados nivel por nivel. En cada nivel, intenta unir los correspondientes estados con aquellos que se encuentran en los niveles anteriores. Para evitar la sobregeneralización, se utilizan como fuentes de conocimiento a los modelos de lenguaje del origen y del destino y un diccionario bilingüe. Los modelos de lenguaje evitan aquellas uniones que darían lugar a frases gramaticalmente incorrectas. El diccionario se utiliza para garantizar que cada palabra destino sea la traducción de una o más palabras de la frase origen (algunas palabras pueden ser la traducción de la palabra vacía).

Un criterio adicional para la mezcla de estados es el hacer uso de modelos de alineamiento estadístico. En particular, se puede utilizar de dichos modelos un diccionario bilingüe y el alineamiento explícito entre pares de frases. Para ello, cada estado se etiqueta con dos conjuntos: las «garantías» y las «necesidades». El primer conjunto indica aquellas palabras que pueden aparecer en la salida desde el momento en el que la correspondiente entrada se ha visto. El segundo conjunto contiene aquellas palabras que deberían aparecer entre las que se generarán en algún momento por aquellas partes que parten de los estados [Vil00].

El SST inferido a partir del corpus etiquetado con clases podría utilizarse para traducir frases origen donde cada una de las palabras se hubiera reemplazado por su clase correspondiente. Sin embargo, como hemos visto, una palabra origen puede pertenecer a más de una clase dependiendo de cómo fueran agrupadas las palabras extendidas de las que forme parte. Por otro lado, la traducción contendría etiquetas de clases que tendrían que reemplazarse por la traducción correspondiente a la palabra origen que la generó. Esto plantea el siguiente problema: si en la frase de salida tuviéramos una misma etiqueta de clase dos o más veces, ¿cuál de las palabras origen fue la que generó cada una de ellas?

Para evitar estos problemas, la solución pasa por la expansión de las clases que forman parte del SST inferido. Para ello, primero se genera para cada clase un autómata trivial y después se expande el SST utilizando dichos autómatas triviales.

4.4.3. Generación de los autómatas de clases

Cada clase de palabras extendidas puede representarse como un autómata trivial. Este autómata trivial está formado por sólo dos estados: un estado inicial y un estado final. Por cada palabra extendida se crea un arco del estado inicial al final. Cada arco toma como entrada la palabra origen de la palabra extendida correspondiente. La salida de cada arco es una regla de sustitución que indica por qué palabra destino se habría de sustituir la etiqueta de la clase correspondiente.

De esta forma, la clase C182 (ver A.2) a la que pertenece la palabra extendida *giving*$_{darnos}$, generaría el autómata trivial mostrado en la figura 4.11. Como se puede ver, las entradas de los arcos son las palabras del idioma origen (p.e. *darnos*) y las salidas corresponden a reglas de sustitución (p.e. *C182=«giving»*). Estas reglas serán utilizadas posteriormente para sustituir en la frase de salida la clase indicada por su correspondiente palabra destino.

Como se ve en 4.11 es posible asignar una probabilidad de transición a cada arco. Esta probabilidad puede estimarse como el número de veces que se ha visto la palabra extendida dividida por el número total de veces que se ha visto la clase. Dichas probabilidades podrían utilizarse para la generación de SSTs estocásticos basados en clases utilizando otros algoritmos en lugar de OMEGA.

Puede ocurrir que varias palabras extendidas compartan una misma palabra origen. En dicho caso, si se generara un autómata trivial en la forma indicada, tendríamos un autómata no determinista. Es decir, habría más de un arco sa-

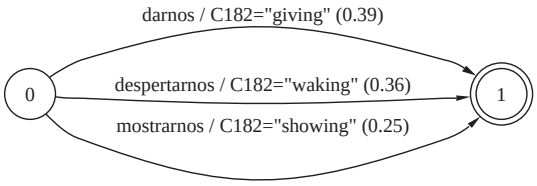


Figura 4.11: Autómata trivial de la clase C182.

liendo del mismo estado con la misma entrada. A modo de ejemplo, la figura 4.12 muestra el autómata trivial correspondiente a la clase C17 (ver A.2) generado según el procedimiento anterior. En dicho autómata se puede observar como las palabras de entrada *llevar*, *llevarme* y *llevarnos* generan dos arcos cada una. Cada arco con una traducción distinta: *carrying* o *sending*.

Las palabras de salida en conflicto pueden ser sinónimos y por lo tanto sería indistinto utilizar una u otra. Pero también puede darse el caso de que, dependiendo del contexto o de la traducción, sea más correcto utilizar una u otra. ¿Cuál de los arcos debería escoger el traductor? Aunque hiciéramos uso de un autómata con probabilidades en los arcos, que no es el caso, esta probabilidad sólo indicaría que un arco es más probable que otro —ha ocurrido más veces— pero no podríamos considerar otros aspectos como el contexto o la frase que se está traduciendo que son los que realmente determinarían la opción más correcta. Habría que posponer esta decisión.

La solución que hemos adoptado es la siguiente. Cuando varias palabras extendidas de la misma clase comparten una misma palabra origen, se genera un sólo arco que tendrá como salida todas las posibles palabras del idioma destino marcadas como *palabras alternativas*. Será el traductor el que, más adelante, cuando haya generado la frase completa con las palabras alternativas en su lugar, generé tantas traducciones como sean necesarias, las puntúe y seleccione la mejor de ellas. En la implementación actual, dicha puntuación se realiza en base a un modelo de traducción entre la frase de entrada y las posibles traducciones alternativas (se puede ver un ejemplo del proceso en la sección 4.5). A modo de ejemplo, la figura 4.13 muestra el autómata trivial determinista correspondiente a la clase C17 (ver A.2). Las palabras alternativas asociadas a las palabras *llevar*, *llevarme* y *llevarnos* están marcadas como tales y aparecen en los correspondientes arcos. La notación empleada ha sido la de colocar entre llaves las palabras alternativas y separarlas con el carácter '|'.

4.4.4. Expansión del SST

El SST generado a partir del corpus etiquetado con clases puede expandirse utilizando los autómatas triviales de clases. Para realizar dicha expansión basta

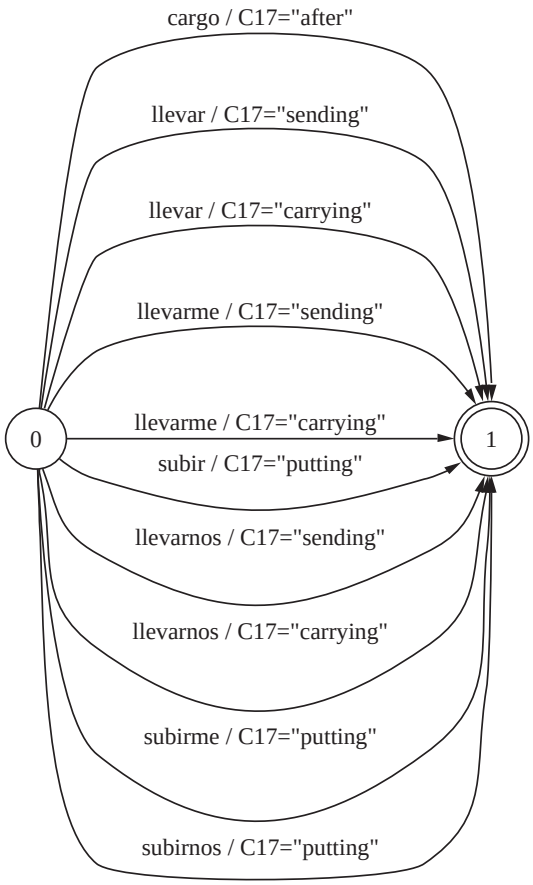


Figura 4.12: Autómata trivial de la clase C17 (no determinista).

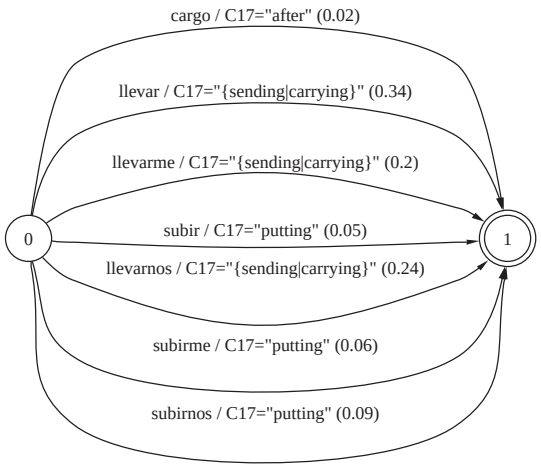


Figura 4.13: Autómata trivial de la clase C17 (determinista).

con sustituir cada arco del SST original en el que aparece como entrada una determinada clase por aquellos arcos correspondientes al autómata trivial de dicha clase. Los nuevos arcos tendrán como entrada la palabra origen y como salida, la salida del arco original precedida de la regla de sustitución correspondiente. De esta forma, se genera un SST que acepta el lenguaje origen y genera frases en el idioma destino. Las frases generadas tendrán, además de palabras en el idioma destino, clases y reglas de sustitución de dichas clases por las palabras destino correspondientes.

A modo de ejemplo, la figura 4.14 muestra una parte de un SST en el que aparecen algunos de los arcos que salen del estado «3» del SST. Como se puede ver, hay tres arcos que tienen como entrada las clases C17, C123 y C182, respectivamente. La figura 4.15 muestra como resultaría la expansión de la parte del SST mostrada en la figura 4.14.

4.5. Traducción con un SST expandido

Como se ha visto en el capítulo 2, un SST es una red de estados finitos determinista que acepta frases de un lenguaje de entrada y produce frases asociadas en un lenguaje de salida. Es decir, en principio, para obtener una traducción una vez entrenado un SST, bastaría con introducir la frase que se quiere traducir y la salida del SST sería la traducción correspondiente.

Sin embargo, los SSTs no pueden traducir aquellas frases de entrada que no cumplan con las reglas sintácticas que éstos impongan. Estas restricciones sintácticas pueden deberse a errores en las frases de entrada o a la falta de

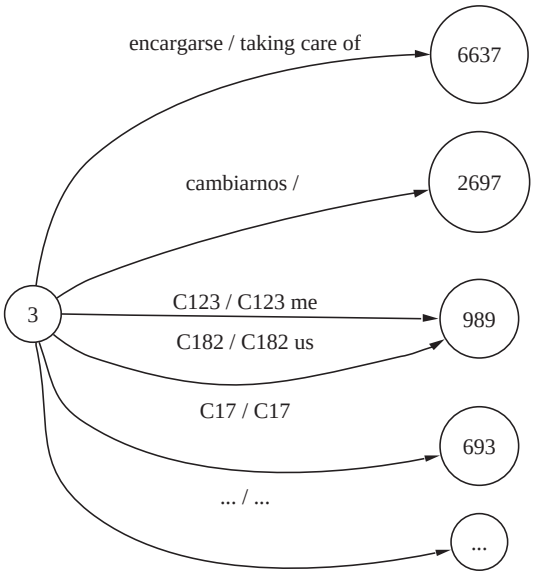


Figura 4.14: Parte de un SST sin expandir.

generalización en el aprendizaje. Los errores en la entrada pueden deberse a variaciones en el vocabulario, repeticiones, desaparición de palabras, palabras superfluas o mal colocadas, etc. La falta de generalización en el entrenamiento puede hacer que se rechacen frases de entrada que sí son correctas con respecto al dominio de traducción.

La utilización de técnicas de análisis de corrección de errores (*error-correcting parsing* [ABC+97b]) proporciona a los SSTs la robustez requerida para superar dichos problemas. Bajo esta aproximación, la frase de entrada x_I se considera una versión distorsionada de alguna frase $x \in L$, siendo L el dominio o lenguaje de entrada del SST. El proceso de distorsión se modela por medio de un modelo de error, E, que tiene en cuenta inserciones, sustituciones y borrados. Se asume que x es aquella frase que minimiza una determinada función de diferencia d_E. La traducción de x_I se obtiene finalmente como la traducción de la mejor estimación de x.

La traducción obtenida no es todavía la traducción final. Puede que como parte de la traducción tengamos etiquetas de clases y las correspondientes reglas de sustitución. El siguiente paso consiste en la sustitución de las etiquetas de clases por su palabra correspondiente. Además, si la sustitución de algunas de las clases incluye palabras alternativas (ver 4.4.3) habrá que generar traducciones alternativas, puntuarlas y seleccionar la mejor.

Veamos el proceso de traducción mediante dos ejemplos. El primero de ellos

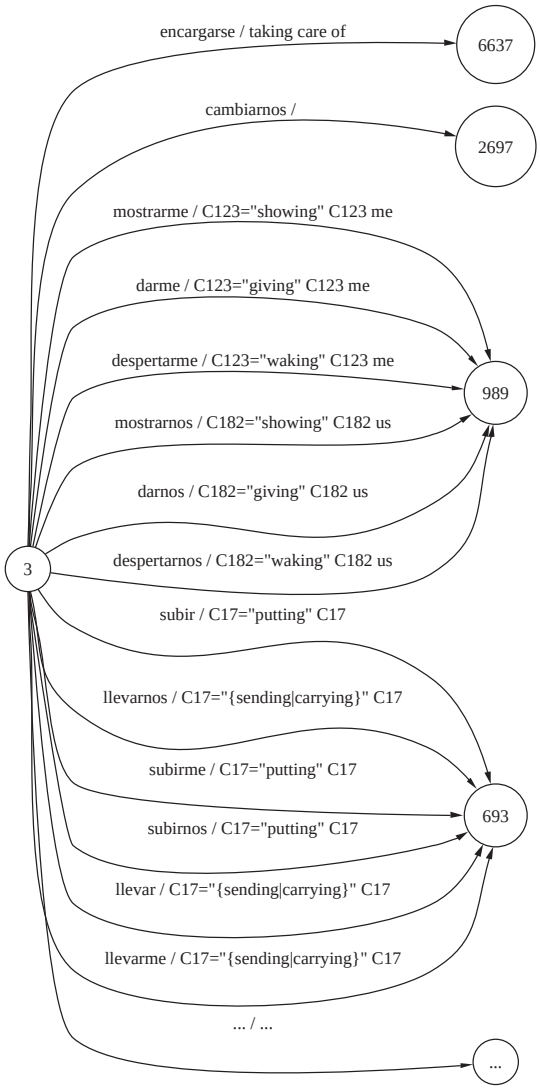


Figura 4.15: Parte de un SST expandido. Ver las figuras 4.11, 4.13 y 4.14.

sin palabras alternativas y el segundo con ellas. Para los ejemplos se ha utilizado un SST expandido entrenado con el corpus EuTrans I (ver capítulo 6) utilizando las clases bilingües mostradas en la sección A.2.

La primera frase de entrada que vamos a ver cómo se traduce es:

por favor , despiértenos mañana a las siete y cuarto .

Utilizando el método de análisis corrector de errores se busca una frase en el dominio de entrada del SST lo más cercana posible a la frase anterior. (Lo ideal sería que el SST hubiera generalizado lo suficiente como para aceptar tal cual la frase de entrada.) La frase encontrada en este caso ha sido:

por favor, despiértenos mañana a las siete y cuarto .

que coincide con la frase que queremos traducir. Hemos comprobado que esta frase no estaba originalmente en el corpus de entrenamiento, es decir, el proceso de entrenamiento ha generalizado lo suficiente como para que el SST acepte dicha frase.

A continuación, se recorre el autómata siguiendo la frase más cercana encontrada. Por cada arco por el que se pasa se genera su salida. La secuencia de salidas forma la siguiente frase:

C145=«wake» please C145 us up tomorrow at C24=«seven» a quarter past C24 .

A continuación, se sustituyen las etiquetas de las clases *C145* y *C24* por las palabras indicadas por las reglas de sustitución *C145=«wake»* y *C24=«seven»*. Como no hay *palabras alternativas*, el proceso acaba aquí y la traducción queda (correctamente):

please wake us up tomorrow at a quarter past seven .

Veamos mediante el siguiente ejemplo que pasaría si hubieran aparecido palabras alternativas en alguna de las reglas de sustitución. Sea la frase de entrada:

¿ les importaría llevarme el equipaje a mi habitación , por favor ?

Al igual que antes, se busca la frase más cercana (que para este ejemplo coincide con la de entrada) y se recorre el SST generando las salidas de los arcos por los que se pasa. La salida será en este caso:

C1=«would» C1 you C92=«mind» C92 C17=«{sending|carrying}» C17 my luggage to my room , please ?

Tras aplicar las reglas de sustitución se llega a:

would you mind {sending|carrying} my luggage to my room , please ?

Puesto que en la posición 4 de la frase han aparecido las palabras alternativas *sending* y *carrying*, se generan las siguientes frases:

would you mind sending my luggage to my room , please ?
would you mind carrying my luggage to my room , please ?

Tras la evaluación de cada una de las frases como posible traducción de la frase original utilizando un modelo de traducción, se otorga una mayor puntuación a la siguiente frase que es la que se presenta (correctamente) como traducción:

would you mind sending my luggage to my room , please ?

4.6. Conclusiones

El concepto de palabras extendidas presentado en este capítulo permite la realización de un agrupamiento bilingüe que, como tal, tiene como objetivo la generación de mejores traductores. Para la realización de este agrupamiento se pueden utilizar los algoritmos vistos en el capítulo anterior. Sólo se habrán de modificar para permitir que ciertas palabras (las que no sean palabras extendidas) sean asignadas cada una a una clase propia de la que no puedan moverse.

Este agrupamiento bilingüe se puede utilizar para la generación de traductores automáticos. Se ha presentado una posible integración de dicho agrupamiento en el modelo de SSTs. El procedimiento consiste en la generación de un SST expandido y es a grandes rasgos el siguiente. El corpus bilingüe se etiqueta con las clases obtenidas. Se entrena un SST con este corpus etiquetado y para cada clase se genera un autómata trivial. Finalmente, el SST se expande utilizando los autómatas triviales.

Este SST expandido puede utilizarse para la traducción de frases del idioma origen. La traducción se obtiene finalmente tras un sencillo post-proceso de la salida generada por el SST.

Capítulo 5

Colocaciones

5.1. Introducción

En el capítulo anterior vimos cómo generar un agrupamiento bilingüe utilizando el alineamiento entre palabras de dos idiomas. El agrupamiento propuesto se basa en la utilización de un modelo de traducción palabra a palabra y genera clases bilingües formadas por pares de palabras en ambos idiomas. En aquellos casos en que una palabra del idioma origen aparece alineada con más de una palabra del idioma destino, esta aproximación escoge sólo una de ellas como su traducción más probable. Sin embargo, determinadas palabras en un idioma se traducen realmente por una secuencia de palabras en otro idioma. Estas secuencias de palabras que constituyen de alguna forma una entidad reciben el nombre de *colocaciones*. La identificación automática de estas secuencias de palabras permitiría generar agrupamientos bilingües entre palabras en el idioma origen y palabras o colocaciones en el idioma destino.

En este capítulo proponemos un método, basado en uno de los de I. Dan Melamed, que identifica colocaciones de forma automática a partir de un corpus bilingüe. Este método encuentra secuencias de palabras en el idioma destino que son traducción de una determinada palabra en el idioma origen. Primero se evalúan secuencias de dos palabras y posteriormente se examinan colocaciones formadas por más palabras partiendo de las colocaciones ya identificadas.

En esta tesis estamos interesados únicamente en la identificación de las colocaciones en el idioma destino. Sin embargo, el procedimiento descrito en este capítulo puede utilizarse para la identificación de colocaciones en ambos idiomas.

En la siguiente sección daremos una definición más formal de colocación y la matizaremos para adaptarla al contexto de esta tesis. La sección 5.3 mues-

tra el método propuesto para la identificación de una colocación. En primer lugar veremos cómo podemos evaluar si una determinada colocación es válida. A continuación, veremos como podemos estimar la idoneidad de una colocación antes de que sea evaluada y por último, cómo podemos limitar el número de secuencias de palabras que probaremos para identificar posibles colocaciones. La sección 5.4 presenta el algoritmo propuesto para la identificación automática de colocaciones. La sección 5.5 muestra cómo generar un agrupamiento bilingüe utilizando dichas colocaciones, cómo entrenar un sistema de traducción automática y el proceso que se sigue para la generación de traducciones. Finalmente, en la sección 5.6 presentamos las conclusiones de este capítulo.

5.2. Conceptos generales de colocaciones

Una posible definición de colocación sería [MS00, pág. 183]:

> [Una colocación se define como] una secuencia de dos o más palabras consecutivas, que presenta características de una unidad sintáctica y semántica, y cuyo significado exacto y no ambiguo no puede desprenderse directamente del significado o connotaciones de sus componentes.

Los siguientes criterios son utilizados generalmente para la identificación manual de colocaciones:

- No composicionalidad. El significado de una colocación no es una composición directa de los significados de sus componentes. El significado o es completamente diferente (caso de las frases hechas) o existe una connotación o un significado añadido que no puede ser inferido desde las partes (p.e. *hace un día de perros*).
- No sustituibilidad. No podemos sustituir los componentes de una colocación por otras palabras aún cuando, en contexto, tengan el mismo significado (p.e. *hace un día de canes*).
- No modificabilidad. Muchas colocaciones no pueden ser libremente modificadas por medio de material léxico adicional o de transformaciones gramaticales (p.e. *hace un día de perros de raza*).

Algunas de las técnicas comúnmente utilizadas para la identificación de colocaciones en corpus monolingües se basan en el cálculo de la frecuencia de las secuencias de palabras en el corpus; la media y varianza de la distancia entre las palabras que se consideran parte de una colocación; en el resultado de determinados tests estadísticos sobre secuencias de palabras y en el cálculo de la información mutua entre palabras. Estas técnicas, sin embargo, están pensadas exclusivamente para la identificación de colocaciones propias a un idioma.

Melamed [Mel97] propone un método diferente para descubrir colocaciones o compuestos no composicionales (*non-compositional compounds*), como él los denomina. Melamed parte de la idea de que una colocación no suele traducirse palabra por palabra en otro idioma. De esta forma, la comparación entre dos idiomas debiera poner de manifiesto aquellas secuencias de palabras que son colocaciones. Dicho método se basa en la comparación de dos modelos estadísticos de traducción inducidos sobre un corpus bilingüe. Uno de los modelos, al que llama *modelo base*, se infiere sobre el corpus original. El segundo modelo, al que llama *modelo modificado*, se infiere sobre el corpus original modificado de la siguiente forma: se ha sustituido cada aparición de la secuencia de palabras que se quiere evaluar por una nueva palabra. Para la inferencia de estos modelos, Melamed utiliza un método de alineamiento entre palabras que relaciona una con una las palabras de ambos idiomas. Es decir, no permite alineamientos de varias palabras a una en ninguno de los dos sentidos. A continuación, calcula la información mutua entre palabras alineadas en ambos modelos. Si la información mutua es mayor en el modelo modificado que en el modelo base, entonces acepta la colocación que estaba evaluando. En otro caso, la rechaza. Siguiendo este procedimiento identifica determinadas secuencias de palabras como colocaciones gracias a la relación entre ambos idiomas. Para reducir el coste computacional propone métodos para evaluar varias colocaciones de forma simultánea y técnicas para estimar la idoneidad de cada colocación. Nosotros utilizaremos también técnicas similares en nuestra propuesta.

En nuestro caso, estamos interesados en la identificación de secuencias de palabras que son traducción de una determinada palabra en el otro idioma, aún cuando dicha secuencia no sea una colocación propiamente dicha. La consideraremos como tal desde el punto de vista de la traducción. Por ejemplo, la palabra *despiértenos* en castellano se traduce por la secuencia de palabras *wake us up*. La expresión *wake us up* no es realmente una colocación en el idioma inglés. Sin embargo, para nosotros, *wake us up* será una colocación traducción de *despiértenos*.

Además, condicionaremos que una determinada secuencia de palabras sea tratada como colocación a las palabras de las que sea traducción. Es decir, una determinada secuencia de palabras podrá ser o no una colocación dependiendo de cuál sea su traducción. Por ejemplo, dados los pares de frases:

¿Le importaría darnos las llaves de la habitación?
Would you mind giving us the keys to the room?

y

¿Nos podría dar las llaves de la habitación?
Would you mind giving us the keys to the room?

Podemos ver como en el primer par de frases, la secuencia *giving us* se produce como traducción de la palabra *darnos*. En el segundo par, *giving* es la traducción de *dar* y *us* de *nos*. Querremos que *giving us* sea considerada una colocación si es traducción de *darnos*, pero no si es la traducción de *dar* o de *nos*. No limitamos esta relación a una sola palabra. Es decir *giving us* podría ser también una colocación en el caso de que fuera debida, por ejemplo, a *entregarnos*.

Un ejemplo que pone de relieve la utilidad de diferenciar en base a la traducción es el siguiente. En castellano, cuando nos referimos al número de una habitación, decimos el número completo; como si fuera una cantidad. Por ejemplo, «*la habitación ciento quince*». Mientras que en inglés, el número se diría cifra a cifra. Por ejemplo, «*room number one one five*». Siguiendo esta traducción, podríamos querer considerar a *one five* como una colocación, ya que la secuencia de palabras *one five* es la traducción de la palabra en castellano *quince*. Ahora bien, si miramos la traducción del número de habitación *ciento cincuenta y dos*, veremos que ésta es *one five two*. Si hubiéramos considerado *one five* como una colocación, es decir, una entidad, negaríamos el hecho de que la palabra *one* es la traducción de *ciento* y la palabra *five* de *cincuenta*. Por lo tanto, estamos interesados en identificar *one five* como una colocación válida, pero sólo si aparece como traducción de *quince*.

Resumiendo, a diferencia de los métodos clásicos de identificación de colocaciones que determinan qué secuencias de palabras de un idioma son colocaciones, proponemos un método que determina qué secuencias de palabras cuando aparecen como traducción de una determinada palabra son colocaciones.

5.3. Identificación de las colocaciones

Comenzamos esta sección definiendo un modelo de traducción sencillo. La estimación de los parámetros de este modelo será distinta dependiendo de que consideremos a una determinada secuencia de palabras como una colocación o no. De esta forma, podremos utilizar la *bondad* de esta estimación para la identificación de las colocaciones.

Sea $\mathrm{P}(x, \bar{y})$ un modelo de traducción sencillo en el que cada palabra del idioma origen genera una secuencia de palabras del idioma destino. Este modelo, si lo utilizáramos para traducir, sólo nos proporcionaría por cada palabra del idioma origen una secuencia de palabras en el idioma destino. No sabría ordenar las palabras propuestas para generar una frase de salida correcta.

Las secuencias de palabras del idioma destino utilizadas por el modelo presentan las siguientes características. Están formadas por palabras que se generan como traducción de una palabra origen y no tienen que ser necesariamente consecutivas. El orden de las palabras dentro de la secuencia indica el orden en el que aparecieron como traducción de la palabra origen. Es decir, consideraremos que dos secuencias son distintas si a pesar de tener las mismas palabras, éstas aparecen en un orden distinto.

En nuestro modelo de traducción sencillo, puede darse el caso de que una palabra del idioma origen no genere ninguna palabra, si es así, diremos que genera una secuencia de palabras formada únicamente por la palabra vacía. Además, consideraremos que cada frase del idioma origen posee una palabra vacía capaz de generar una secuencia de palabras del idioma destino; aquellas que no han sido generadas por ninguna palabra de la frase origen.

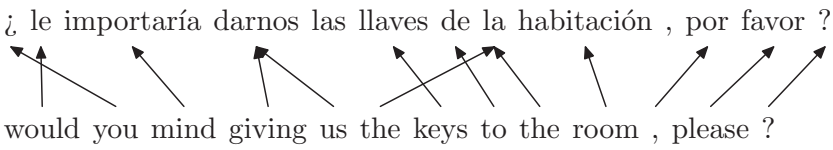


Figura 5.1: Ejemplo de frases alineadas con el modelo 2 de IBM.

Por ejemplo, para el alineamiento generado por el modelo 2 de IBM del par de frases representado en la figura 5.1 obtendríamos los pares de palabra del idioma origen y secuencia del idioma destino mostrados en siguiente cuadro:

x	$\bar{y}$	x	$\bar{y}$
λ	λ	de	to
¿	you	la	the the
le	would	habitación	room
importaría	mind	,	λ
darnos	giving us	por	,
las	λ	favor	please
llaves	keys	?	?

El par (λ,λ) indica que la palabra vacía de la frase origen genera la secuencia formada por la palabra vacía; todas las palabras del idioma destino están alineadas con alguna palabra del idioma origen. El siguiente par relaciona la palabra origen «¿» con la secuencia *you*. El siguiente, *le* con *would* y así sucesivamente. Las palabras origen *las* y «,» que no están alineadas con ninguna palabra del idioma destino generan los pares (*las*, λ) y («,» , λ). Además, el par (*la, the the*) muestra que palabras no consecutivas pueden formar una secuencia.

Dado un corpus bilingüe de entrenamiento podemos estimar el modelo $\mathrm{P}(x, \bar{y})$. Para ello, basta con obtener el alineamiento entre palabras y utilizar las cuentas entre palabras alineadas. Usando la estimación por máxima-verosimilitud tendríamos:

$$\mathrm{P}(x, \bar{y}) = \frac{\mathrm{n}(x, \bar{y})}{N}, \tag{5.1}$$

donde $\mathrm{n}(x, \bar{y})$ es el número de veces que la secuencia $\bar{y}$ ha sido alineada con x, y N es el número de alineamientos existentes entre palabras del idioma origen y secuencias del idioma destino, lo que en la forma en la que hemos definido el modelo equivale al número de palabras existentes en el idioma origen del corpus de entrenamiento (más una palabra vacía por frase).

Podemos considerar que un par $(x, \bar{y})$ es una instancia del par de variables aleatorias X e $\bar{Y}$ que representan a las posibles palabras origen y a las posibles secuencias de palabras destino. De esta forma, podemos utilizar la entropía conjunta, $\mathrm{H}(X, \bar{Y})$, de las variables aleatorias X e $\bar{Y}$ para cuantificar la cantidad de

información requerida como media para especificar ambos valores. La definición de $\mathrm{H}(X, \bar{Y})$ es la siguiente:

$$\mathrm{H}(X, \bar{Y}) = -\sum_{x \in X} \sum_{\bar{y} \in \bar{Y}} \mathrm{P}(x, \bar{y}) \cdot \log \mathrm{P}(x, \bar{y}). \tag{5.2}$$

Cuanto mejor sea la estimación realizada de $\mathrm{P}(x, \bar{y})$ menor será la entropía conjunta. Dicha estimación depende de qué secuencias de palabras están alineadas contra qué palabras del idioma origen. Por tanto, para probar si una (sub)secuencia de palabras $y_1 \dots y_n$ es una colocación cuando aparece como traducción de una palabra x, bastaría con entrenar dos modelos de traducción, uno sin considerar la secuencia como colocación y otro considerándola como tal, y comparar las respectivas entropías conjuntas. El primero de los modelos, al que denominaremos *modelo base*, correspondería al de la distribución original. El segundo de los modelos, al que denominaremos *modelo modificado*, consideraría a la secuencia $y_1 \dots y_n$ en aquellos casos en los que en el modelo original alguna de sus palabras estuviera alineada con x, como una sola palabra desde el punto de vista del alineamiento. De esta forma, las palabras, $y_1 \dots y_n$, acabarían alineadas con una sola palabra en cada frase en la que antes alguna de ellas, pero no necesariamente todas, estaba alineada con x. Si cuando calculamos la entropía conjunta del modelo modificado, ésta es menor o igual a la del modelo base, entonces consideraremos que la secuencia de palabras $y_1 \dots y_n$ como traducción de x es una colocación válida.

Por ejemplo, supongamos que queramos evaluar *giving us* como colocación traducción de *darnos*. En primer lugar obtendríamos el alineamiento para el corpus bilingüe original. Estimaríamos el modelo base de traducción $\mathrm{P}(x, \bar{y})$ y calcularíamos su $\mathrm{H}(X, \bar{Y})$. A continuación, modificaríamos el corpus original de forma que cada vez que *giving* y *us* aparezcan consecutivamente en la misma frase estando al menos una de ellas alineada con *darnos*, se reemplazara la secuencia *giving us* por una sola palabra; por ejemplo, *giving_us*. Sobre este nuevo corpus obtendríamos el nuevo alineamiento. A continuación, estimaríamos el modelo modificado de traducción $\mathrm{P}'(x, \bar{y})$ y calcularíamos su $\mathrm{H}'(X, \bar{Y})$. Si $\mathrm{H}'(X, \bar{Y}) \leq \mathrm{H}(X, \bar{Y})$ aceptaríamos a *giving us* como colocación traducción de *darnos*.

Conviene matizar que las palabras *compuestas* en el idioma destino (p.e. *giving_us*) aparecen como palabras independientes en las secuencias de palabras generadas. Por ejemplo, si *darnos* aparece alineado con la palabra compuesta *giving_us*, se generaría el par (*darnos*, *giving us*).

Si observamos el proceso descrito, podemos observar que el modelo modificado se diferencia del modelo base únicamente en la nueva distribución de $\bar{Y}$, la distribución de X permanece constante. Habrán cambiado algunas secuencias de palabras $\bar{y}$, pero las palabras x del idioma origen seguirán siendo las mismas. Como $\mathrm{H}(X)$ permanece constante entre modelos y $\mathrm{H}(X, \bar{Y}) = \mathrm{H}(\bar{Y} \mid X) + \mathrm{H}(X)$, podemos utilizar la mejora en la entropía de $\bar{Y}$ dada X, $\mathrm{H}(\bar{Y} \mid X)$, entre los modelos base y modificado para la identificación de las colocaciones. $\mathrm{H}(\bar{Y} \mid X)$

mide la cantidad de información necesaria para especificar $\bar{Y}$ si se conoce X. Se calcula como:

$$\mathrm{H}(\bar{Y} \mid X) = -\sum_{x \in X} \sum_{\bar{y} \in \bar{Y}} \mathrm{P}(x, \bar{y}) \cdot \log \mathrm{P}(\bar{y} \mid x), \tag{5.3}$$

donde $\mathrm{P}(\bar{y} \mid x)$ se estima como $\frac{\mathrm{n}(x,\bar{y})}{\mathrm{n}(x)}$.

La generación y evaluación de cada nuevo modelo es una operación costosa. Por tanto, no parece conveniente utilizar el procedimiento descrito anteriormente para probar una a una todas las posibles colocaciones. Sería interesante poder evaluar varias colocaciones de forma simultánea con cada nuevo modelo generado. Además, puesto que deberemos probar un gran número de colocaciones, la estimación de la mejora de $\mathrm{H}(\bar{Y} \mid X)$ debida a una posible colocación permitiría descartar colocaciones no válidas sin necesidad de evaluarlas en un nuevo modelo. También permitiría establecer la prioridad de aquellas que sí han de ser evaluadas. Otra cuestión que no hemos tratado hasta ahora es qué secuencias de palabras son posibles colocaciones. ¿Cualquier secuencia de palabras del idioma destino? En las siguientes subsecciones trataremos estos puntos. Primero mostramos cómo podemos evaluar varias colocaciones de forma simultánea. En la subsección 5.3.2 presentamos el método propuesto para estimar la aportación de cada colocación a la mejora del modelo. Por último, en la subsección 5.3.3 veremos una forma de restringir el número de secuencias que evaluaremos como posibles colocaciones.

5.3.1. Evaluación simultánea de varias colocaciones

Para evaluar de forma simultánea varias colocaciones será necesario determinar cuál es la contribución de cada posible colocación a la diferencia entre $\mathrm{H}'(\bar{Y} \mid X)$ y $\mathrm{H}(\bar{Y} \mid X)$. De esta forma, podremos ver qué términos corresponden a cada colocación y evitar, en la medida de lo posible, que se solapen los términos de las distintas colocaciones.

Para simplificar el cálculo de la contribución de cada colocación, se considerarán inicialmente aquellas formadas por secuencias de sólo dos palabras. El algoritmo mostrado en la sección 5.4 identificará colocaciones formadas por más palabras de forma iterativa.

Para calcular la aportación de la colocación $y_1 y_2$ como traducción de x_c al incremento de $\mathrm{H}(\bar{Y} \mid X)$ realizaremos la siguiente suposición:

Suposición 5.1. *Si se genera un modelo modificado en el que las palabras y_1 e y_2 se consideran una posible colocación traducción de x_c, $\mathrm{P}(x, \bar{y})$ sólo variará para aquellas x, además de x_c, que estén originalmente alineadas con y_1 ó y_2 cuando ocurren conjuntamente y alguna de ellas está alineada con x_c.*

Si denominamos c a la colocación $y_1 y_2$ como traducción de x_c y X_c al subconjunto de X que incluye a x_c y aquellas x alineadas con y_1 o y_2 cuando ocurren conjuntamente y alguna de ellas estaba originalmente alineada con x_c,

podríamos usar la suposición 5.1 para calcular el incremento de $\mathrm{H}(\bar{Y} \mid X)$ debido a la colocación c como:

$$\begin{aligned}\Delta\mathrm{H}(\bar{Y} \mid X) &= \mathrm{H}'(\bar{Y} \mid X) - \mathrm{H}(\bar{Y} \mid X) \\ &\simeq -\sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \mathrm{P}'(x, \bar{y}) \log \mathrm{P}'(\bar{y} \mid x) \\ &\quad + \sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \mathrm{P}(x, \bar{y}) \log \mathrm{P}(\bar{y} \mid x). \end{aligned} \tag{5.4}$$

Condición de exclusión mutua. A la vista de (5.4) podemos decir que para poder evaluar simultáneamente n colocaciones, $c_1 \ldots c_n$, debemos garantizar, como mínimo, que $\cap_{i=0}^{n} X_{c_i} = \emptyset$. En caso contrario, no podríamos separar la aportación individual de cada colocación. Desde un punto de vista práctico, consideraremos que la palabra vacía no interviene en la condición.

5.3.2. Estimación de la contribución de cada colocación

El número de colocaciones que pueden comprobarse simultáneamente está limitado por la anterior condición de exclusión mutua. Sería deseable, por tanto, poder determinar a priori las colocaciones más prometedoras. De esta forma, para aquellas colocaciones que se excluyan mutuamente, podríamos evaluar en primer lugar aquellas que tengan más posibilidades de ser aceptadas. Esta estimación también debería permitirnos rechazar posibles colocaciones sin necesidad de evaluarlas.

Expresando (5.4) en función de las cuentas $\mathrm{n}(x, \bar{y})$, el incremento de $\mathrm{H}(\bar{Y} \mid X)$ debido a una colocación c vendría dado por:

$$\begin{aligned}\Delta\mathrm{H}(\bar{Y} \mid X) &\simeq -\sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \frac{\mathrm{n}'(x, \bar{y})}{N'} \log \frac{\mathrm{n}'(x, \bar{y})}{\mathrm{n}'(x)} \\ &\quad + \sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \frac{\mathrm{n}(x, \bar{y})}{N} \log \frac{\mathrm{n}(x, \bar{y})}{\mathrm{n}(x)}. \end{aligned} \tag{5.5}$$

Para calcular la estimación de dicho incremento será necesario estimar los valores de $\mathrm{n}'(x)$, N' y $\mathrm{n}'(x, \bar{y})$. El número de veces que una palabra origen aparece como parte de un par palabra origen secuencia de palabras destino no depende de si juntamos o no determinadas palabras destino. Por lo tanto, $\mathrm{n}'(x)$ será igual a $\mathrm{n}(x)$. Por otro lado, N' es el nuevo número de pares de palabras origen y secuencias de palabras destino. Puesto que para cada frase, cada palabra origen —incluyendo una palabra vacía— estará alineada con una secuencia de palabras destino —que puede estar formada por la palabra vacía— independientemente de cómo variemos $\bar{Y}$, el número de pares alineados seguirá siendo el mismo. Es decir, $N' = N$. Por lo tanto, sólo será necesario estimar $\mathrm{n}'(x, \bar{y})$.

Para poder estimar $\mathrm{n}'(x, \bar{y})$ dada la colocación $y_1 y_2$ como traducción de x_c realizaremos las siguientes suposiciones:

Suposición 5.2. *Sólo se modificaran en el nuevo modelo aquellas secuencias $\bar{y}$ que contenían a y_1 o a y_2 pero no a ambas. Además, estas secuencias se modificaran de la siguiente forma. Si $\bar{y}$ estaba alineado con x_c, entonces se incorporará a ella, la palabra y_1 ó y_2 que falte. Si $\bar{y}$ no estaba alineado con x_c, entonces se suprimirá de ella, la palabra y_1 ó y_2 que forme parte de $\bar{y}$.*

Suposición 5.3. *Las nuevas secuencias $\bar{y}$ continuarán estando alineadas con la palabra x con la que estaban alineadas originalmente.*

Llamamos $\hat{\mathrm{n}}(x, \bar{y})$ a la estimación de $\mathrm{n}'(x, \bar{y})$. Utilizando las anteriores suposiciones, podemos calcular $\hat{\mathrm{n}}(x, \bar{y})$ a partir de $\mathrm{n}(x, \bar{y})$ y de las variaciones esperadas de $\bar{Y}$. Para poder expresar $\hat{\mathrm{n}}(x, \bar{y})$, comenzaremos realizando una serie de definiciones. Éstas se aplican en el contexto de un par de frases, una traducción de la otra. Como hemos visto, $\bar{y}$ es una determinada secuencia ordenada de palabras destino alineadas con una palabra origen x. Llamaremos $\bar{y}\,|_y$ a la secuencia de palabras de la que forma parte la palabra y en el par de frases actual. También definiremos las operaciones de *suma* y de *resta* sobre una secuencia $\bar{y}$ como sigue. La operación $\bar{y} + y$ inserta la palabra y en la secuencia $\bar{y}$ en la posición que le corresponde de acuerdo al orden en el que aparecen las palabras en la frase actual del idioma destino. La operación $\bar{y} - y$ elimina la palabra y de la secuencia $\bar{y}$. Por último y abusando de la notación, definiremos una función de alineamiento, $\vec{a}(\cdot)$, de la siguiente forma: $\vec{a}(y)$ devuelve la palabra x con la que la palabra y está alineada en el par de frases actual.

Proponemos el algoritmo 5.1 para calcular $\hat{\mathrm{n}}(x, \bar{y})$ debida a la colocación c ($y_1 y_2$ como traducción de x_c). Éste requiere un corpus bilingüe, una función de alineamiento entre palabras obtenida para dicho corpus y los valores actuales de $\mathrm{n}(x, \bar{y})$. Comienza asignando $\mathrm{n}(x, \bar{y})$ a $\hat{\mathrm{n}}(x, \bar{y})$ y creando una lista de pares $(x, \bar{y})$, inicialmente vacía, en la que se almacenarán aquellos pares $(x, \bar{y})$ que sean modificados debido a la adopción de la colocación.

El algoritmo procede de la siguiente forma. Para cada aparición del bigrama $y_1 y_2$ en el corpus, calcula las variaciones de $n(x, \bar{y})$, $d(x, \bar{y})$, debidas a que c sea tratada como una colocación. Que c sea tratada como una colocación implica que las palabras y_1 e y_2 en aquellas ocasiones en las que al menos una de las dos estuviera alineada originalmente con x_c actuarán como una palabra *compuesta*. Debido a esto, variará la distribución de $\bar{Y}$ y $\mathrm{n}(x, \bar{y})$. El algoritmo calcula estas variaciones de acuerdo con las suposiciones 5.2 y 5.3 incrementando o decrementando convenientemente $d(x, \bar{y})$. Por último, una vez recorridos todas las apariciones de $y_1 y_2$ en el corpus, se actualiza $\hat{\mathrm{n}}(x, \bar{y})$ con las diferencias calculadas para la lista de pares modificados.

Finalmente, dada $\hat{\mathrm{n}}(x, \bar{y})$, podemos estimar el incremento de $\mathrm{H}(\bar{Y} \mid X)$ debido

Algoritmo 5.1 Obtención de $\hat{\mathrm{n}}(x, \bar{y})$ debido a la posible colocación $(x_c, y_1 y_2)$

Requiere: $\mathcal{C}_{xy}$, $\vec{a}(\cdot)$, $\mathrm{n}(\cdot)$
  Inicializar $\hat{\mathrm{n}}(x, \bar{y}) \leftarrow \mathrm{n}(x, \bar{y})$, $ListaPares \leftarrow \emptyset$
  para cada aparición del bigrama $y_1 y_2$ en el corpus $\mathcal{C}_{xy}$ **hacer**
    si $\vec{a}(y_1) \neq \vec{a}(y_2) \wedge \big(\vec{a}(y_1) = x_c \vee \vec{a}(y_2) = x_c\big)$ **entonces**
      si $\vec{a}(y_1) = x_c$ **entonces**
        $y_a \leftarrow y_1$; $y_b \leftarrow y_2$
      si no
        $y_a \leftarrow y_2$; $y_b \leftarrow y_1$
      fin
      // Cambios debidos a la palabra y_a (alineada con x_c)
      $ListaPares \leftarrow ListaPares + (x_c, \bar{y} \mid_{y_a})$
      $ListaPares \leftarrow ListaPares + (x_c, \bar{y} \mid_{y_a} + y_b)$
      $d(x_c, \bar{y} \mid_{y_a}) \leftarrow d(x_c, \bar{y} \mid_{y_a}) - 1$
      $d(x_c, \bar{y} \mid_{y_a} + y_b) \leftarrow d(x_c, \bar{y} \mid_{y_a} + y_b) + 1$
      // Cambios debidos a la palabra y_b (no alineada con x_c)
      $ListaPares \leftarrow ListaPares + (\vec{a}(y_b), \bar{y} \mid_{y_b})$
      $ListaPares \leftarrow ListaPares + (\vec{a}(y_b), \bar{y} \mid_{y_b} - y_b)$
      $d(\vec{a}(y_b), \bar{y} \mid_{y_b}) \leftarrow d(\vec{a}(y_b), \bar{y} \mid_{y_b}) - 1$
      $d(\vec{a}(y_b), \bar{y} \mid_{y_b} - y_b) \leftarrow d(\vec{a}(y_b), \bar{y} \mid_{y_b} - y_b) + 1$
    fin
  fin para
  para cada $(x, \bar{y})$ en $ListaPares$ **hacer**
    $\hat{\mathrm{n}}(x, \bar{y}) \leftarrow \mathrm{n}(x, \bar{y}) + d(x, \bar{y})$
  fin para

a una colocación c como:

$$\begin{aligned} \hat{\Delta}\mathrm{H}(\bar{Y} \mid X) \;=\; & -\sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \frac{\hat{\mathrm{n}}(x, \bar{y})}{N} \log \frac{\hat{\mathrm{n}}(x, \bar{y})}{\mathrm{n}(x)} \\ & + \sum_{x \in X_c} \sum_{\bar{y} \in \bar{Y}} \frac{\mathrm{n}(x, \bar{y})}{N} \log \frac{\mathrm{n}(x, \bar{y})}{\mathrm{n}(x)}. \end{aligned} \tag{5.6}$$

5.3.3. Selección de las posibles colocaciones

Queda por plantear cuáles son las posibles colocaciones que debemos probar. Puesto que el procedimiento descrito comienza considerando colocaciones formadas por dos palabras, una posible opción sería la de probar todos los bigramas que aparecen el corpus del idioma destino. Esta opción conllevaría probar una gran cantidad de posibles colocaciones.

Para limitar el número de posibles colocaciones que deban ser evaluadas, proponemos la siguiente restricción. Utilizando un modelo de alineamiento de varias

palabras a una, podemos seleccionar como posibles colocaciones sólo aquellos bigramas y_1y_2 en los que, en al menos una ocasión, tanto y_1 como y_2 han sido alineados contra una misma palabra x. De esta forma, limitamos la búsqueda a aquellos pares de palabras que el modelo de alineamiento sugiere como posibles colocaciones.

5.4. Algoritmo de identificación automática de colocaciones

El algoritmo propuesto para la identificación automática de colocaciones en un corpus bilingüe, $\mathcal{C}_{xy}$, es el siguiente:

1. Inicializar una *lista de colocaciones no válidas* y una *lista de colocaciones válidas*.
2. Obtener un modelo de traducción base, $\mathrm{P}(x, \bar{y})$, para $\mathcal{C}_{xy}$.
3. Calcular el $\hat{\Delta}\mathrm{H}(\bar{Y} \mid X)$ para aquellos bigramas y_1y_2 en $\mathcal{C}_y$ y las palabras x en $\mathcal{C}_x$ que no estén en la *lista de colocaciones no válidas* y tales que y_1 e y_2 estén alineados simultáneamente al menos en una frase con x.
4. Generar una *lista de colocaciones candidatas* que contenga aquellos bigramas para los que $\hat{\Delta}\mathrm{H}(\bar{Y} \mid X) \leq 0$ ordenados de menor a mayor $\hat{\Delta}\mathrm{H}(\bar{Y} \mid X)$.
5. Borrar de la *lista de colocaciones candidatas* aquellas c que no satisfagan la condición de exclusión mutua con alguna c' situada antes en la lista.
6. Obtener $\mathcal{C}'_{xy}$ sustituyendo cada una de las colocaciones candidatas por una palabra compuesta que la represente.
7. Obtener un modelo de traducción, $\mathrm{P}'(x, \bar{y})$, para $\mathcal{C}'_{xy}$.
8. Calcular $\Delta\mathrm{H}(\bar{Y} \mid X)$ para cada colocación candidata.
9. Para cada colocación candidata, $c = (x, y_1y_2)$, tal que $\Delta\mathrm{H}(\bar{Y} \mid X) \mid_c \leq 0$, añadir c a la *lista de colocaciones válidas*, en caso contrario, añadirla a la *lista de colocaciones no válidas*.
10. Sustituir en $\mathcal{C}_{xy}$ cada colocación de la *lista de colocaciones válidas* por una palabra compuesta que la represente.
11. Si quedan colocaciones candidatas por evaluar, volver al paso 2.

Como resultado obtendremos una lista de colocaciones aceptadas y un nuevo corpus bilingüe, $\mathcal{C}_{xy}$, en el cada secuencia de palabras del idioma destino que forma una colocación ha sido reemplazada por una nueva palabra que la representa. Por ejemplo, si la secuencia *giving us* se ha identificado como colocación

traducción de *darnos*, cada vez que en el corpus aparecía la secuencia *giving us* donde alguna de sus palabras había sido alineada con *darnos*, en el nuevo corpus tendríamos una nueva palabra que representaría a dicha colocación, por ejemplo *giving_us*.

Conviene tener en cuenta que el algoritmo descrito obtiene colocaciones en el idioma destino, pero puede modificarse fácilmente para que en cada iteración se busquen alternativamente colocaciones en el idioma destino y en el idioma origen. La condición de parada sería entonces la no existencia de colocaciones candidatas por evaluar en ninguno de los idiomas.

5.5. Agrupamiento bilingüe e inferencia de SSTs con colocaciones

Si se dispone de un corpus bilingüe en el que se ha reemplazado cada colocación detectada por una palabra que la representa, podemos realizar un agrupamiento bilingüe sobre dicho corpus utilizando el método descrito en el capítulo anterior. Para el algoritmo de agrupamiento bilingüe las palabras compuestas serán palabras sin más. La diferencia entre utilizar o no colocaciones es la siguiente. Si no utilizamos colocaciones, el método propuesto de agrupamiento bilingüe agruparía tan sólo a una de las palabras que forman la colocación. Utilizando colocaciones, éstas aparecen como una única palabra y serán agrupadas como un todo. Es de esperar, por tanto, que el agrupamiento bilingüe utilizando colocaciones sea de mayor calidad.

La generación de un traductor basado en clases con colocaciones se puede realizar de la forma descrita en la sección 4.4. Para que el traductor así generado tenga sentido, la identificación de colocaciones debe realizarse únicamente en el idioma destino. De esta forma, el traductor aceptará frases del idioma origen y generará frases del idioma destino (con alguna que otra palabra compuesta que habrá que procesar). El problema que presenta la identificación de colocaciones en ambos sentidos es que el traductor necesitaría que se le suministrara una frase de entrada con las colocaciones adecuadas. Puesto que estamos relacionando las colocaciones con las palabras de las que son traducción, podemos generar una colocación habiendo visto la entrada. Sin embargo, no podemos decidir si las palabras de la frase de entrada forman alguna colocación puesto que no hemos visto la traducción. Al menos, no de forma trivial. Esta opción se plantea como una de las mejoras futuras de esta tesis.

Mostramos a continuación el proceso de traducción con un par de frases de ejemplo (comparar con la sección 4.5). El primero de ellos sin palabras alternativas y el segundo con ellas. Para los ejemplos se ha utilizado un SST expandido entrenado con el corpus EuTrans I (ver capítulo 6) utilizando las clases bilingües con colocaciones mostradas en la sección A.4.

Veamos cómo se traduciría la frase:

por favor , despiértenos mañana a las siete y cuarto .

En primer lugar y utilizando el método de análisis corrector de errores se busca una frase en el dominio de entrada del SST lo más cercana posible a la frase anterior. La frase encontrada en este caso coincide con la original. A continuación, se recorre el autómata siguiendo dicha frase. Por cada arco por el que se pasa se genera su salida. La secuencia de salidas forma la siguiente frase:

C37=«wake␣us␣up» please C37 tomorrow at C3=«seven» a quarter␣past C3 .

Como podemos ver, han aparecido en la salida dos colocaciones: *wake␣us␣up* que ha aparecido como traducción de *despiértenos* y *quarter␣past* como traducción de la palabra *cuarto* en la expresión *y cuarto*.

El siguiente paso para la obtención de la traducción es la sustitución de las etiquetas de las clases *C37* y *C3* por las palabras indicadas por las reglas de sustitución *C37=«wake␣us␣up»* y *C3=«seven»*. La traducción incluyendo colocaciones queda:

please wake␣us␣up tomorrow at a quarter␣past seven .

El último paso es trivial y consiste en eliminar los caracteres «␣», con lo que la traducción queda correctamente:

please wake us up tomorrow at a quarter past seven .

Otro ejemplo, pero esta vez con posibles alternativas se produce al traducir la frase:

¿ nos podría subir la bolsa de viaje a la habitación número cuatro doce , por favor ?

De nuevo la frase más cercana coincide con la de entrada. Y la salida generada por el autómata después de recorrer dicha frase es:

C1=«{could|can}» C1 you send up C50=«{my|our}» C50 travel bag to room C3=«four» number C3 C63=«one␣two» C63 , please ?

Cuando aplicamos las reglas de sustitución obtenemos la frase múltiple:

{could|can} you send up {my|our} travel bag to room number four one␣two , please ?

Expandiendo esta frase, obtenemos las siguientes cuatro posibilidades:

could you send up my travel bag to room number four one␣two , please ?
could you send up our travel bag to room number four one␣two , please ?
can you send up my travel bag to room number four one␣two , please ?
can you send up our travel bag to room number four one␣two , please ?

Tras la evaluación de cada una de las frases como posible traducción de la frase original utilizando un modelo de traducción, se otorga una mayor puntuación a la siguiente frase que es la que se presenta como traducción:

could you send up my travel bag to room number four one␣two , please ?

La traducción generada no es correcta debido a que en lugar de la palabra *my* debería haber aparecido la palabra *our*. Sin embargo, la frase correcta:

could you send up our travel bag to room number four one␣two , please ?

sí que estaba entre las propuestas, es decir, mejorando la fase de puntuación de las traducciones alternativas podríamos mejorar las traducciones generadas. El problema de este caso en concreto se debe a que tanto *my* como *our* han sido alineadas con la palabra *la* (de *la bolsa de viaje*) y no se ha capturado la relación de éstas con las palabras *me* y *nos*, respectivamente.

Al igual que en el ejemplo anterior, el último paso consiste en eliminar los caracteres «␣», con lo que la traducción queda:

could you send up my travel bag to room number four one two , please ?

5.6. Conclusiones

En este capítulo hemos mostrado un procedimiento que detecta colocaciones de forma automática en un corpus bilingüe. Hemos visto cómo evaluar varias posibles colocaciones de forma simultánea para reducir el coste del proceso de evaluación. Sin embargo, sólo pueden evaluarse de forma simultánea aquellas colocaciones que satisfagan una condición que hemos denominado de *exclusión mutua*. Debido a esto, hemos desarrollado un procedimiento que estima la idoneidad de cada posible colocación. De esta forma, si varias colocaciones no satisfacen la condición de exclusión mutua podemos evaluar primero aquella cuya estimación sea mejor. Además, esta estimación nos permite descartar directamente aquellas posibles colocaciones cuya estimación indique que no son válidas.

Finalmente, hemos descrito la integración de las colocaciones en el proceso de aprendizaje de un sistema de traducción basado en clases. Y el proceso seguido para llevar a cabo una traducción utilizando dicho sistema.

En el capítulo 6 presentamos resultados experimentales utilizando colocaciones para la obtención de agrupamientos bilingües y en el apéndice A se pueden comparar dos agrupamientos del corpus EuTrans I: el primero de ellos sin colocaciones y el segundo utilizando colocaciones. Además, en ese apéndice también se muestran las colocaciones detectadas de forma automática en el corpus EuTrans I.

Capítulo 6

Experimentos

6.1. Introducción

En este capítulo presentamos una serie de experimentos que tienen por objeto comprobar el funcionamiento de los métodos propuestos de agrupamiento monolingüe, agrupamiento bilingüe y colocaciones. Los experimentos realizados con agrupamiento monolingüe comparan los métodos presentados en esta tesis entre sí y con los métodos propuestos en [BDd+92] y [Och99]. Mientras que los experimentos realizados con agrupamiento bilingüe y colocaciones se centran en el efecto que estos métodos tienen sobre la mejora de sistemas de traducción automática; en concreto, la mejora de transductores subsecuenciales. Para la realización de los experimentos hemos utilizado los corpora bilingües EuTrans I y EuTrans II. Comenzamos este capítulo describiendo ambos corpora.

EuTrans I es un corpus bilingüe castellano–inglés formado por 10.000 pares de frases de entrenamiento y 2.996 de test. Su vocabulario es de 686 palabras en castellano y de 513 en inglés. El cuadro 6.1 muestra un resumen de sus principales características. Este corpus abarca una serie de situaciones típicas de comunicación entre un turista y el recepcionista de un hotel como son [ABC+97a]:

- Pedir habitaciones, despertador, llaves, la cuenta, un taxi y mover el equipaje.
- Preguntar acerca de las habitaciones (disponibilidad, características, precio...).
- Mirar habitaciones, quejarse y cambiar de habitación.

		Castellano	Inglés
Entrenamiento	Pares de frases	10.000	
	Pares de frases distintas	6.813	
	Palabras	132.198	134.922
	Vocabulario	686	513
	Longitud media de frase	9,7	9,9
	Longitud máxima	30	30
	Perplejidad (bigramas)	8,6	6,3
Test	Pares de frases (todas distintas)	2.996	
	Longitud media de frase	11,7	11,9
	Longitud máxima	31	30
	Vocabulario	611	468
	Palabras fuera de vocabulario	–	–

Cuadro 6.1: Principales características del corpus EuTrans I.

— ¿ le importaría darnos las llaves de la habitación , por favor ? — would you mind giving us the keys to the room , please ?
— por favor , ¿ podrían subirnos la maleta a la nueve dieciséis ? — could you send up our suitcase to room number nine one six , please ?
— desearía dos habitaciones dobles y tranquilas . — I would like two quiet , double rooms .

Cuadro 6.2: Ejemplos de pares de frases del corpus EuTrans I.

- Notificar una reserva previa.
- Rellenar la hoja de registro.
- Pedir y quejarse sobre la cuenta.
- Avisar de la marcha.
- Otras expresiones comunes.

Algunos pares de frases de dicho corpus se muestran a modo de ejemplo en el cuadro 6.2.

El segundo de los corpora utilizados, el corpus EuTrans II (v 5.1) [CNO+], se divide también en una parte de entrenamiento y otra de test. La parte de entrenamiento está formada por 3.038 pares de frases, una traducción de la otra, en italiano e inglés. Su vocabulario es de 2.534 palabras en italiano y 1.701 en inglés. El cuadro 6.3 muestra un resumen de las características de dicho corpus.

		Italiano	Inglés
Entrenamiento	Pares de frases (todas distintas)	3.038	
	Palabras	61.232	72.446
	Vocabulario	2.534	1.701
	Longitud media de frase	17,9	21,5
	Longitud máxima	81	94
	Perplejidad (bigramas)	27	15
Test	Pares de frases	300	
	Pares de frases distintas	296	
	Longitud media de frase	20,1	24,1
	Longitud máxima	59	67
	Vocabulario	715	547
	Palabras fuera de vocabulario	107	66

Cuadro 6.3: Principales características del corpus EuTrans II.

La tarea que abarca el corpus EuTrans II es bastante más compleja y cercana a la realidad que la del corpus EuTrans I. Para la generación de este corpus se almacenaron una serie de llamadas de teléfono en italiano a la recepción de un hotel. La recogida de las llamadas se hizo de forma automática y se utilizaron técnicas de Mago de Oz [ACDC99] para simular conversaciones reales. El corpus recogido es bastante espontáneo.

Para generar la versión en texto de este corpus, se transcribieron las frases almacenadas y se tradujeron al inglés. De este corpus, 3.308 pares de frases se han utilizado para el entrenamiento de los modelos y 300 pares de frases se han reservado para el test. El cuadro 6.4 muestra algunos pares de frases del corpus EuTrans II.

El resto del capítulo está organizado como sigue. En la siguiente sección se presentan los experimentos de agrupamiento monolingüe. En la sección 6.3 se muestran los experimentos de agrupamiento bilingüe y colocaciones para la mejora de sistemas de traducción automática.

La mayor parte de los resultados mostrados en este capítulo se presentan en forma de gráficas. Los resultados numéricos se pueden consultar en el apéndice B. Para poder localizar fácilmente los resultados numéricos, en el título de las figuras se indica el número de página en el que se pueden encontrar.

6.2. Agrupamiento monolingüe

Para la realización de los experimentos presentados en esta sección hemos utilizado las siguientes aplicaciones de agrupamiento monolingüe:

- `ngram-class`. Forma parte del *SRI Language Modeling Toolkit*. Se puede

— Excelsior ? buon giorno vorrei prenotare una stanza tripla , per il periodo dal ventitré al ventinove dicembre con servizi in camera , frigorifero e televisore , grazie . — Excelsior ? good morning I would like to reserve a triple room , for the period from twenty-third to the twenty-ninth of December with services in room , minibar and television set , thanks .
— volevo disdire la camera a nome di Maccarrone . grazie . — I would like to cancel the room in the name of Maccarrone . thank you .
— salve , sto chiamando dalla stazione dei treni . vorrei posticipare una prenotazione effettuata per il mese di dicembre , dal ventitré al ventinove e spostarla , se é possibile , nel mese di gennaio . grazie . — hello , I am calling from the railway station . I would like to postpone the reservation made for the month of December , from the twenty-third to the twenty-ninth and to move it , if it is possible , to the month of January . thank you .

Cuadro 6.4: Ejemplos de pares de frases del corpus EuTrans II.

descargar de:
`http://www.speech.sri.com/projects/srilm/`

- `mkcls`. Desarrollado por Franz Josef Och. Se puede descargar de:
  `http://www-i6.informatik.rwth-aachen.de/~och/software/mkcls.html`

- `ecluster++`. Desarrollado por Sergio Barrachina. Implementa los métodos de agrupamiento descritos en esta tesis.

Utilizando estas aplicaciones hemos generado agrupamientos con los siguientes métodos de agrupamiento monolingüe:

- `srilm_inc`. Las clases se construyen utilizando un algoritmo voraz incremental. Comienza con una clase para cada una de las C palabras más frecuentes y añade una palabra cada vez. Esta es una implementación del algoritmo de $O(V \cdot C^2)$, donde V es el tamaño del vocabulario, descrito en [BDd$^+$92]. Corresponde a la opción `-incremental` del programa `ngram-class`.

- `srilm_full`. Las clases se construyen utilizando un algoritmo voraz sobre todas las clases comenzando por una clase por palabra. Es una implementación del algoritmo de $O(V^3)$ descrito en [BDd$^+$92]. Corresponde a la opción `-full` del programa `ngram-class`.

- `mkcls`. El método implementado se describe en [Och99]. Se ha utilizado la siguiente línea de comandos:

`mkcls -cC -n10 -pin -Vout opt`,
donde C es el número de clases.

- `iterativo`. Corresponde al algoritmo *iterativo* propuesto en esta tesis y descrito en la sección 3.4. Realiza una distribución inicial de las palabras en C clases e iterativamente mueve palabras de unas clases a otras buscando un mejor agrupamiento.

- `iterativo_lo`. Corresponde al algoritmo *iterativo dejando uno fuera* propuesto en esta tesis y descrito en la sección 3.5. El modo de operar es idéntico al método anterior salvo por la optimización que se realiza utilizando la técnica de *dejar uno fuera*.

- `incremental_lo`. Corresponde al algoritmo *incremental* propuesto en esta tesis y descrito en la sección 3.6. Como método de optimización se utiliza el descrito en la sección 3.5 basado en la técnica de dejar uno fuera.

En las siguientes subsecciones se muestran los experimentos de agrupamiento monolingüe realizados sobre los corpora EuTrans I y EuTrans II.

6.2.1. Experimentos con EuTrans I

Para la realización de los experimentos de agrupamiento monolingüe sobre el corpus EuTrans I hemos utilizado la parte en castellano de dicho corpus.

Experimentos preliminares. Los métodos `iterativo_lo` e `incremental_lo` requieren que se fije el valor del parámetro b (ver sección 3.5). Por ello, se ha realizado un experimento preliminar utilizando el método incremental dejando uno fuera con $b = 0{,}75$ [KN93] con distintos números de clases entre 100 y 686. Para cada número de clases se ha calculado la estimación del parámetro b óptimo (ver ecuación 3.24). Los valores obtenidos se muestran en el cuadro 6.5. La media de estos valores, 0,4, se ha utilizado como valor de b para los restantes experimentos.

Experimento 1. Con este experimento pretendemos comparar la *calidad* de los agrupamientos generados sobre el corpus EuTrans I por los distintos métodos de agrupamiento. Para ello, hemos procedido de la siguiente forma. Utilizando cada uno de los métodos de agrupamiento monolingüe, se han realizado agrupamientos con distintos números de clases del corpus de entrenamiento. Posteriormente, se ha entrenado un modelo de lenguaje basado en clases utilizando el corpus de entrenamiento y cada uno de los agrupamientos generados. Finalmente, se ha obtenido la perplejidad medida sobre el corpus de test para cada modelo de lenguaje. Cuanto menor sea esta perplejidad, mejor será el modelo y por tanto, mejor será el método de agrupamiento utilizado por dicho modelo. Es

Clases	b	Clases	b
100	0,43	450	0,38
150	0,35	500	0,47
200	0,33	550	0,54
250	0,33	600	0,62
300	0,31	650	0,66
350	0,30	686	0,61
400	0,35		

Cuadro 6.5: Estimación del parámetro b para distintos números de clases del idioma castellano del corpus EuTrans I.

decir, utilizaremos la perplejidad sobre el test como una medida de la calidad de los agrupamientos generados.

Los resultados obtenidos se presentan en la figura 6.1. En ella se muestra la perplejidad medida sobre el test para el modelo de bigramas y la variación de ésta con el número de clases para los métodos de agrupamiento comparados. Se han realizado agrupamientos con distintos números de clases entre 100 y 686 (donde 686 es la talla del vocabulario de la parte en castellano del corpus EuTrans I).

Como se puede observar, la perplejidad de los modelos de lenguaje basados en los distintos agrupamientos desciende desde las 100 clases hasta encontrar un mínimo entre las 200 y 250 clases. Y luego asciende hasta que, para el máximo número de clases posibles (tantas clases como palabras del vocabulario), alcanza la perplejidad del modelo de bigramas.

El método `incremental_lo`, propuesto en esta tesis, alcanza el mejor resultado de todos los métodos comparados. Otro comportamiento que habría que destacar de dicho método es la evolución de su perplejidad con el número de clases es mucho más suave que la del resto de métodos; asciende lentamente desde las 250 hasta las 686 clases, mientras que el resto de métodos experimenta una fuerte subida entre las 250 y 300 clases. Como se puede observar, a partir de las 250 clases la perplejidad obtenida por este método de agrupamiento se mantiene siempre por debajo de la del resto de métodos analizados.

Los métodos `iterativo` e `iterativo_lo`, también propuestos en esta tesis, y los métodos `srilm_inc`, `srilm_full` y `mkcls` obtienen perplejidades similares. Dependiendo del número de clases, uno u otro de los métodos obtiene una menor perplejidad. Puesto que la variación de perplejidad obtenida por todos estos métodos es muy pequeña podemos concluir que los agrupamientos obtenidos serán similares.

Finalmente, si se comparan entre sí los métodos `iterativo` e `iterativo_lo`, se puede observar que el segundo obtiene una mejor perplejidad en la mayoría de los casos. Por lo que es de suponer que obtiene mejores agrupamientos.

Como ejemplo del tipo de agrupamientos generados, presentamos en el cua-

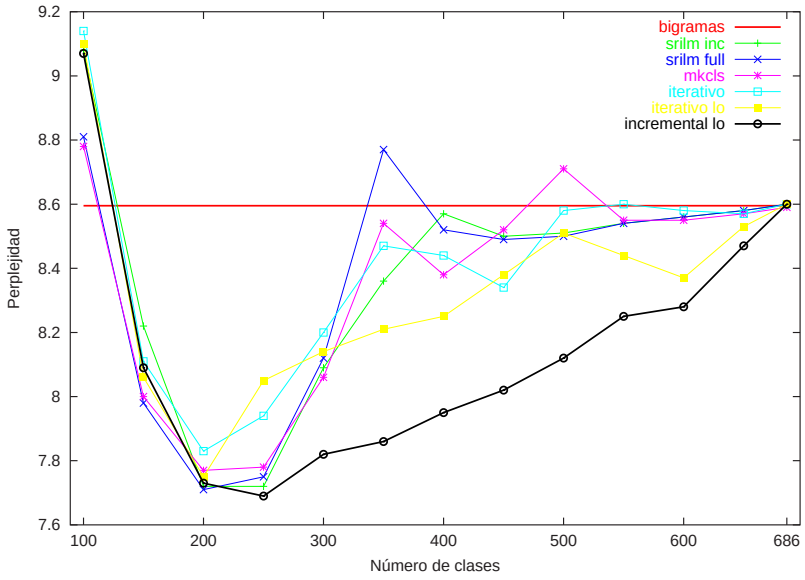


Figura 6.1: Perplejidad sobre el test del idioma castellano del corpus EuTrans I para bigramas y para distintos modelos de clases con números de clases entre 100 y 686. (pág. 184)

dro 6.6 algunas de las clases generadas por el agrupamiento en 250 clases de la parte castellana del corpus EuTrans I con el método `incremental_lo`.

En las 8 primeras filas de dicho cuadro se puede observar cómo han quedado repartidos los números que aparecen en el corpus de entrenamiento. Las unidades han quedado repartidas en tres clases. Los números del *diez* al *veintinueve* han quedado en dos clases: *diez*, *once* y *doce* por un lado, y del *trece* al *veintinueve* por otro. Las decenas salvo el *veinte* en dos clases y las centenas todas en la misma clase. Un aspecto positivo del agrupamiento realizado es que en estas clases no aparecen otras palabras que no sean números.

La explicación de por qué los números aparecen en distintas clases viene dada por su utilización en el texto de entrenamiento. Así, nos podemos encontrar con números que forman parte de fechas (días del mes y horas) y con números que forman parte de números de habitaciones. Las palabras del *uno* al *doce* se utilizan para representar horas, días de la semana y números de habitaciones. Las palabras del *trece* al *treinta* (*treinta y uno*, son tres palabras) se utilizan para representar días del mes y números de habitaciones. Las centenas sólo se utilizan para indicar números de habitaciones. Por último, la palabra *dos*, aparece además en el contexto de *reservar dos habitaciones* (en el que no aparece ningún otro número de los vistos —aparece *una*, como en *reservar una habitación*—).

1:	uno
2:	dos
3:	tres, cuatro, cinco, seis, siete, ocho, nueve
4:	diez, once, doce
5:	trece, catorce, quince, dieciséis, diecisiete, dieciocho, diecinueve, veinte, veintiuno, veintidós, veintitrés, veinticuatro, veinticinco, veintiséis, veintisiete, veintiocho, veintinueve
6:	treinta
7:	cuarenta, cincuenta, sesenta, setenta, ochenta, noventa
8:	ciento, doscientos, trescientos, cuatrocientos, quinientos, seiscientos, setecientos, ochocientos, novecientos
9:	enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre
10:	bosque, lago, mar, río
11:	cuál, cuándo, cuánta, cuántas, cuántos, dónde, quién
12:	adiós, conforme, correcto, disculpe, hola, perdón, perdone, sí

Cuadro 6.6: Ejemplos de clases generadas por el método `incremental_lo` con 250 clases en la parte castellana del corpus EuTrans I.

En la fila 9 del cuadro 6.6 se puede comprobar que todos los meses del año, y sólo los meses del año, han quedado agrupados en la misma clase. En la fila 10 se muestra cómo se han agrupado en la misma clase los pronombres interrogativos que aparecían en el corpus de entrenamiento. En la fila 11 aparecen las palabras *bosque*, *lago*, *mar* y *río* que son los accidentes geográficos contemplados en el corpus de entrenamiento. Las palabras mostradas en la fila 12 aparecen en el corpus de entrenamiento en forma de frases con una sola palabra (a veces, entre signos de exclamación).

Como se ha podido comprobar en este experimento, el método propuesto de agrupamiento monolingüe `incremental_lo` genera agrupamientos de mejor calidad que el resto de métodos comparados y presenta una mayor independencia con el número de clases elegido para el agrupamiento. Además, la inspección de los agrupamientos generados muestra que la mayoría de palabras acaban formando clases razonables. Algunas de estas clases, en las que se pueden observar agrupamientos tanto sintácticos como semánticos, se han presentado en el cuadro 6.6.

Experimento 2. Los métodos de agrupamiento requieren habitualmente que se especifique de antemano el número de clases del agrupamiento. Este experimento tiene por objeto verificar si los métodos propuestos en esta tesis pueden determinar de forma automática el número óptimo de clases. Como se explicó en el capítulo 3, los métodos propuestos utilizan como función de optimización la perplejidad medida sobre el corpus de entrenamiento. Los métodos `iterativo_lo`

e `incremental_lo`, al contrario que el método `iterativo`, utilizan la técnica de dejar uno fuero para simular eventos no vistos e intentar evitar el sobreentrenamiento. El objetivo del experimento es estudiar la posibilidad de utilizar la perplejidad medida sobre el entrenamiento para cada uno de estos métodos para determinar el número óptimo de clases. Para ello, hemos realizado agrupamientos con distintos números de clases entre 100 y 686 del corpus EuTrans I con cada uno de los métodos.

La figura 6.2 representa la evolución con el número de clases de la perplejidad medida sobre el entrenamiento para cada uno de los métodos de agrupamiento. Como se puede observar, la perplejidad medida sobre el entrenamiento del método `iterativo` siempre disminuye conforme aumenta el número de clases hasta alcanzar la de bigramas. Por el contrario, las perplejidades de los métodos `iterativo_lo` e `incremental_lo` descienden inicialmente hasta que alcanzan un mínimo y luego vuelven a ascender. Los mínimos alcanzados por los métodos `iterativo_lo` e `incremental_lo` están en las 300 y 350 clases, respectivamente. Observando de nuevo la figura 6.1, se puede comprobar que el número de clases para el que se obtenía la mejor perplejidad medida sobre el test era de 250.

Por lo tanto, a la vista de los resultados de este experimento es factible utilizar la perplejidad sobre el entrenamiento proporcionada por los métodos `iterativo_lo` e `incremental_lo`, para obtener una aproximación razonable del número óptimo de clases. En la sección 6.3 realizamos más experimentos en los que se corrobora este hecho.

6.2.2. Experimentos con EuTrans II

Para la realización de los experimentos de agrupamiento monolingüe sobre el corpus EuTrans II hemos utilizado la parte en inglés de dicho corpus.

Experimentos preliminares. Como ya vimos en la subsección anterior, los métodos `iterativo_lo` e `incremental_lo` requieren que se fije el valor del parámetro b (ver sección 3.5). Por ello, al igual que antes, se ha realizado un experimento preliminar utilizando el método incremental dejando uno fuera con $b = 0{,}75$ [KN93] y distintos números de clases entre 200 y 1.701 clases. Para cada número de clases se ha calculado la estimación del parámetro b óptimo (ver ecuación 3.24). Los valores obtenidos se muestran en el cuadro 6.7. La media de estos valores, 0,6, se ha utilizado como valor de b en los restantes experimentos.

Experimento 1. Con este experimento pretendemos comparar la *calidad* de los agrupamientos generados sobre el corpus EuTrans II por los distintos métodos de agrupamiento. Para ello, hemos procedido de igual forma que en el experimento 1 de la subsección anterior. Utilizando cada uno de los métodos de agrupamiento monolingüe, se han realizado agrupamientos con distintos números de clases del corpus de entrenamiento. Posteriormente, se ha entrenado un modelo

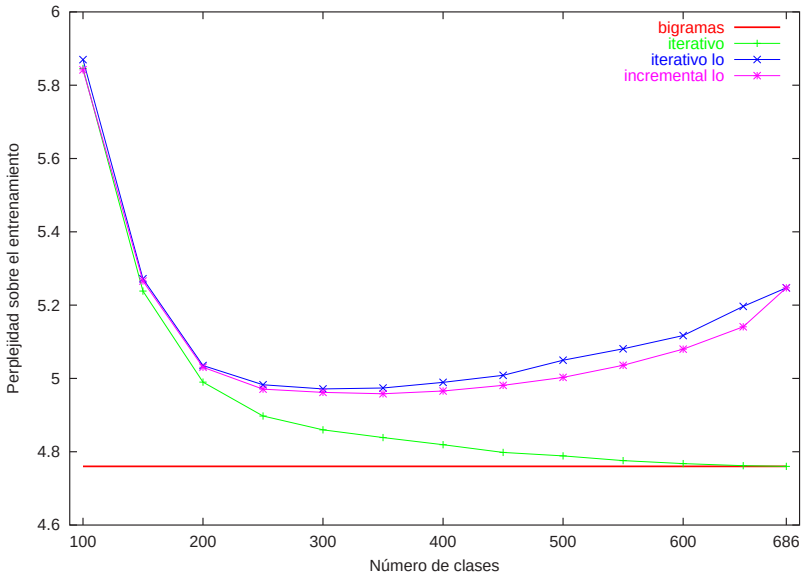


Figura 6.2: Perplejidad sobre el entrenamiento del idioma castellano del corpus EuTrans I para bigramas y para distintos modelos de clases con números de clases entre 100 y 686. (✍ pág. 185)

Clases	b	Clases	b
200	0,67	1.200	0,57
400	0,64	1.400	0,56
600	0,59	1.600	0,61
800	0,57	1.700	0,66
1.000	0,55	1.701	0,66

Cuadro 6.7: Estimación del parámetro b para distintos números de clases del idioma inglés del corpus EuTrans II.

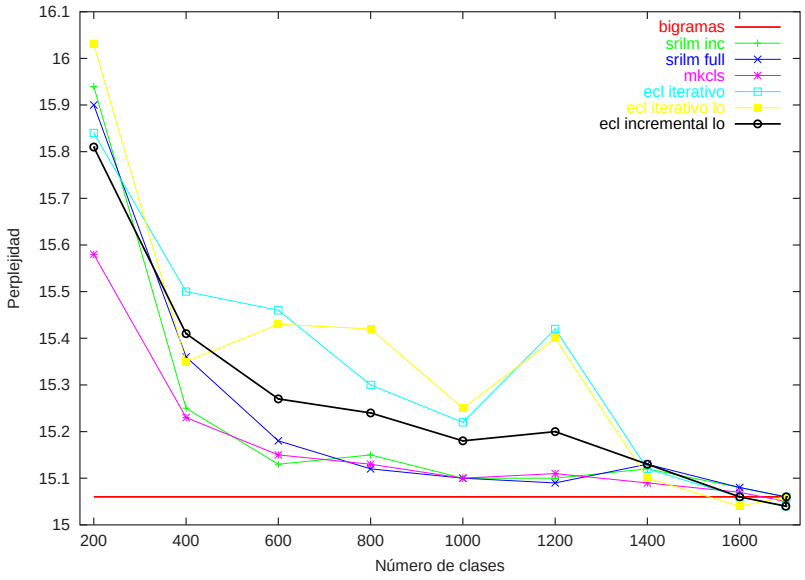


Figura 6.3: Perplejidad sobre el test del idioma inglés del corpus EuTrans II para bigramas y para distintos modelos de clases con números de clases entre 200 y 1.701. (✍ pág. 185)

de lenguaje basado en clases utilizando el corpus de entrenamiento y cada uno de los agrupamientos generados. Finalmente, se ha obtenido la perplejidad medida sobre el corpus de test para cada modelo de lenguaje. Cuanto menor sea esta perplejidad, mejor será el modelo y por tanto, mejor será el método con el que se había entrenado dicho modelo. Es decir, utilizaremos de nuevo la perplejidad sobre el test como una medida de la calidad de los agrupamientos generados.

Los resultados obtenidos se presentan en la figura 6.3. En ella se muestra la perplejidad medida sobre el test para el modelo de bigramas y la variación de ésta con el número de clases para los métodos de agrupamiento comparados. Se han realizado agrupamientos con distintos números de clases entre 200 y 1.701 (donde 1.701 es la talla del vocabulario de la parte en inglés del corpus EuTrans II).

Como se puede observar, las perplejidades obtenidas utilizando los modelos basados en clases casi siempre están por encima de la perplejidad del modelo de bigramas. Algunos de los métodos obtienen una perplejidad ligeramente menor para 1.400, 1.600 y 1.700 clases. Sin embargo, la diferencia es tan pequeña que puede considerarse que no existe una mejora clara sobre el modelo de bigramas.

Por otro lado, la variación de la perplejidad obtenida por los distintos métodos son mínimas; la mayor diferencia entre el mejor y el peor caso para cualquier

número de clases es de 0,45 (además para 200 clases). Los métodos `srilm_inc`, `srilm_full`, `mkcls` e `incremental_lo` obtienen resultados casi idénticos, mientras que los métodos `iterativo` e `iterativo_lo` producen unos resultados ligeramente peores.

Consideramos que estos resultados se deben a que el corpus es demasiado complejo para el número de frases de entrenamiento disponibles. Por tanto, los métodos de agrupamiento comparados no son capaces de reducir la perplejidad del modelo de bigramas sin clases. Dicho de otra forma, el agrupamiento no puede mejorar el conocimiento del lenguaje representado por el corpus de entrenamiento. Creemos que sería necesario disponer de más ejemplos de entrenamiento.

Examinando las clases generadas por los distintos métodos de agrupamiento hemos podido constatar como éstas están en consonancia con los malos resultados mostrados en la anterior figura. Palabras que podrían estar agrupadas en una misma clase acaban en clases distintas y palabras que no guardan demasiada relación entre si acaban en la misma clase. Así, por ejemplo, los nombres propios acaban repartidos en varias clases; los números también aparecen en distintas clases —muy repartidos entre demasiadas clases—, e incluso a veces junto con palabras que no expresan información numérica, etcétera. El cuadro 6.8 muestra algunas clases extraídas al azar del agrupamiento con 400 clases generado por el método `incremental_lo`. En dicho cuadro 6.8 se muestra entre paréntesis, al lado de cada palabra, el número de veces que ha sido vista en el corpus. Como se puede observar, la mayoría de las palabras que aparecen en el cuadro 6.8 se ha visto una única vez en el corpus. De hecho, de las 1701 palabras que forman el vocabulario de la parte inglesa del corpus EuTrans II, 710 de ellas (el 40 %) aparecen una única vez. También es interesante constatar que el número de frases de entrenamiento no llega al doble del vocabulario. Todo esto refuerza la idea de que el entrenamiento disponible es insuficiente para capturar la complejidad de dicho corpus. Además, este corpus contiene una serie de errores tipográficos que dificultan aún más la tarea; por ejemplo, aparece *anylonger* en lugar de *any longer* en más de una ocasión —*any longer* también aparece—, *centre* en lugar de *center*, *tomorrom* en lugar de *tomorrow*...

Experimento 2. Los métodos de agrupamiento requieren habitualmente que se especifique de antemano el número de clases del agrupamiento. Este experimento tiene por objeto verificar si los métodos propuestos en esta tesis pueden determinar de forma automática el número óptimo de clases. Como se expuso en el capítulo 3, los métodos propuestos utilizan como función de optimización la perplejidad medida sobre el corpus de entrenamiento. Los métodos `iterativo_lo` e `incremental_lo`, al contrario del método `iterativo`, utilizan la técnica de dejar uno fuero para simular eventos no vistos e intentar evitar el sobreentrenamiento. El objetivo del experimento es verificar si podemos utilizar la perplejidad medida sobre el entrenamiento por cada uno de estos métodos para determinar el

50:	aerial (1), airport (150), baptistery (3) beach (1), booklet (1), boy (1), centre (5), comforts (1), dates (1), eighth (1), fifth (1), floor (5), harbour (3), island (1), library (1), management (2), manager (3), port (4), sea (1), ship (1), sixth (1), station (88), transfer (1), twentninth (1)
100:	appreciated (5), attend (1), clear (2), particularly (1), retired (1), sufficiently (2), written (1)
150:	decide (2), indicate (4), provide (42), remember (2), understand (10)
200:	past (32)
250:	beds (58)
300:	Chianti (1), anybody (1), back (36), down (14), urgently (9)
350:	went (5)
400:	Alberto (1), agreement (1), bagages (3), baggage (1), crying (3), daughter (1), house (1)

Cuadro 6.8: Ejemplos de clases generadas por el método `incremental_lo` con 400 clases en la parte inglesa del corpus EuTrans II.

número óptimo de clases. Para ello, hemos realizado agrupamientos con entre 200 y 1.701 clases de la parte inglesa del corpus EuTrans II con cada uno de los métodos.

La figura 6.4 representa la perplejidad medida sobre el entrenamiento por cada uno de los métodos para los números de clases entre 200 y 1.701. El método `iterativo` alcanza el mínimo de perplejidad para el máximo número de clases, mientras que los métodos basados en la técnica de dejar uno fuera, `iterativo_lo` e `incremental_lo` descienden hasta 400 clases para luego ascender. Según esta gráfica y las conclusiones expuestas en el experimento 2 de la subsección anterior, podríamos decir que el número estimado de clases óptimo sería por tanto de 400. Sin embargo, puesto que, al menos aparentemente, el número de muestras de entrenamiento no es suficiente, no podemos dar demasiado crédito a este dato ni compararlo con el verdadero óptimo ya que no hay tal (a no ser el de igual número de clases que de palabras). En todo caso, el cuadro 6.8, presentado anteriormente, muestra algunas de las clases extraídas al azar utilizando el método `incremental_lo` con 400 clases.

6.3. Agrupamiento bilingüe y colocaciones

En esta sección se muestran una serie de experimentos en los que se aplican los algoritmos de agrupamiento bilingüe (capítulo 4) y de identificación de colocaciones (capítulo 5) para la inferencia de mejores traductores automáticos.

Los experimentos se han realizado utilizando los corpora EuTrans I y II (descritos en la sección 6.1). La parte de entrenamiento de dichos corpora se

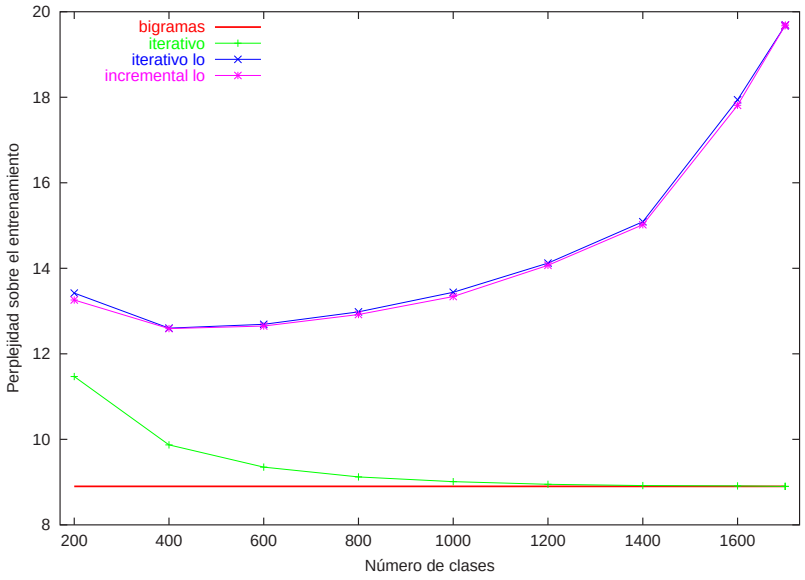


Figura 6.4: Perplejidad sobre el entrenamiento del idioma inglés del corpus EuTrans II para bigramas y para distintos modelos de clases con números de clases entre 200 a 1.701. (pág. 186)

ha utilizado tanto para el agrupamiento bilingüe, como para la identificación de colocaciones y para la inferencia de los traductores automáticos. La parte de test, formada por frases en el idioma origen y sus correspondientes traducciones, se ha utilizado para evaluar la calidad de los traductores inferidos. La evaluación de la calidad de los traductores automáticos se realiza comparando la traducción proporcionada por el test, llamada habitualmente *traducción de referencia*, con la proporcionada por el traductor de forma automática a partir de las frases en el idioma origen.

Para cuantificar la calidad de las traducciones generadas se han utilizado las siguientes medidas:

- BLEU (*Bilingual Evaluation Understudy*). Es un método automático de evaluación de traducciones rápido, barato e independiente del lenguaje. Está fuertemente correlado con la forma de evaluar humana [PRWZ01]. Los valores numéricos que proporciona van desde 0 (traducción pésima) hasta 1 (traducción idéntica a la esperada). Permite utilizar más de una traducción de referencia (sinónimos, distintas formas de expresar la misma idea...). Cuantas más traducciones de referencia, mejor puede ser el valor de la medida. Los corpora utilizados sólo presentan una única traducción de referencia. Podríamos esperar mejorar esta medida si dispusiéramos de

traducciones alternativas.

- WER (*Word Error Rate*). Es la tasa de error de palabras entre la traducción generada y la traducción de referencia. Se calcula como el mínimo número de operaciones de sustitución, inserción y borrado que tienen que realizarse para convertir la traducción generada en la traducción de referencia. Se expresa en tanto por cien.

- PER (*Position-independent Word Error Rate*). Tasa de error propuesta como alternativa al WER. Esta medida compara las palabras de las dos frases (traducción y referencia) sin tener en cuenta el orden de las palabras. Las palabras que no tienen su contrapartida en la otra frase se contabilizan como errores de sustitución. Dependiendo de si la traducción generada es mayor o menor que la traducción de referencia, las restantes palabras se consideran errores de inserción o de borrado que se añaden a los de sustitución ya contabilizados. El PER será siempre menor o igual que el WER [OTN99]. Se expresa en tanto por cien.

- SER (*Sentence Error Rate*). Tasa de error de frases. Expresa en tanto por cien cuántas de las frases generadas difieren de las frases de referencia.

En esta sección mostraremos únicamente los valores de BLEU y WER obtenidos. Y generalmente en forma de gráficas. En el apéndice B mostramos los resultados numéricos, tanto de BLEU, como de WER y del resto de medidas.

Los siguientes procedimientos son comunes a los experimentos presentados en esta sección.

Hemos realizado los alineamientos utilizando alguno de los siguientes modelos y sus correspondientes aplicaciones:

- Modelo 2 de IBM. Hemos utilizado la aplicación `IBMm2` de Juan Miguel Vilar para realizar los alineamientos correspondientes al modelo 2 de IBM. Los experimentos indican que utilizan este modelo mediante la leyenda `IBMm2`.

- Modelo 2 de IBM más vecinos. Hemos utilizado el algoritmo 4.1 (*neighbours*) propuesto en la sección 4.3.1 junto con la aplicación `IBMm2`. Este método surgió para intentar solucionar algunas de las deficiencias observadas experimentalmente en los alineamientos generados por el modelo 2 de IBM. Los experimentos indican que utilizan este modelo mediante la leyenda `IBMm2_nb`.

- Modelos 3 y 4 de IBM. Hemos utilizado la aplicación `GIZA++`[1] [ON00] para realizar los alineamientos correspondientes a los modelos 3 y 4 de IBM. Los experimentos indican que utilizan alguno de estos modelos mediante las leyendas `IBMm3` e `IBMm4`, respectivamente.

[1] `http://www-i6.informatik.rwth-aachen.de/~och/software/GIZA++.html`

Presentamos resultados con los alineamientos `IBMm2` y `IBMm2_nb` principalmente para que, además de comparar resultados con distintos modelos de alineamiento, se puedan comparar los resultados presentados en esta tesis con el trabajo previo del autor en el que se utilizaron estos modelos de alineamiento.

Como sistema de traducción automática hemos utilizado el modelo de transductores subsecuenciales (ver sección 2.5). La inferencia de los transductores subsecuenciales se ha realizado utilizando el algoritmo OMEGA (ver sección 4.4.2). Utilizamos OMEGA —o OMEGA sin clases— para referirnos al traductor generado por este algoritmo.

Cuando el sistema de traducción utiliza información de clases en su entrenamiento nos referiremos a este traductor como «OMEGA más clases». El proceso seguido para la obtención de un traductor de este tipo se describe en la sección 4.4.

Los métodos de agrupamiento bilingües serán los métodos *iterativo*, *iterativo dejando uno fuera* e *incremental dejando uno fuera* (descritos en el capítulo 3) y modificados convenientemente para el caso bilingüe tal y como se describe en el capítulo 4. Nos referiremos a ellos de forma abreviada como `iterativo`, `iterativo_lo` e `incremental_lo`, respectivamente.

Cuando además de la información de clases, utilicemos la información sobre colocaciones para el entrenamiento de un traductor, nos referiremos a este traductor como «OMEGA más clases y colocaciones». El proceso seguido para la obtención de un traductor de este tipo se describe en la sección 5.5.

La generación de las traducciones se ha realizado utilizando las técnicas de corrección de errores descritas en la sección 4.5.

6.3.1. Experimentos con EuTrans I

Experimentos preliminares

El primero de los dos experimentos preliminares realizados tiene por objeto determinar el número de clases adecuado para el modelo 4 de IBM. Este modelo de alineamiento requiere que los lenguajes origen y destino sean agrupados previamente para la generación del alineamiento. Junto con la aplicación `GIZA++` se proporciona la aplicación `mkcls` la generación de estos agrupamientos. En lugar de esta aplicación podríamos haber optado por usar alguno de los métodos de agrupamiento monolingüe propuestos en esta tesis. Sin embargo, puesto que el objetivo de los experimentos realizados en esta sección es la evaluación de los métodos de agrupamiento bilingüe, se ha preferido utilizar el procedimiento estándar para la generación de los agrupamientos monolingües necesarios para el modelo 4 de IBM.

El número de clases utilizado para el agrupamiento de los lenguajes origen y destino se ha determinado realizando el siguiente experimento preliminar. Para cada posible par de número de clases se ha entrenado un SST con el alineamiento generado utilizando dichos agrupamientos. Este SST se ha utilizado para

traducir el test. Por último, se han seleccionado como óptimos los números de clases del lenguaje origen y del lenguaje destino que han proporcionado la mejor traducción. Al proceder de esta forma, las técnicas de agrupamiento bilingüe propuestas en esta tesis utilizarán el mismo alineamiento que el que conduce a la mejor traducción posible cuando no se utilizan.

La figura 6.5 muestra la variación de BLEU y WER con el número de clases de entrada y salida. La figura 6.6 muestra la variación del mejor BLEU y WER con el número de clases de entrada.

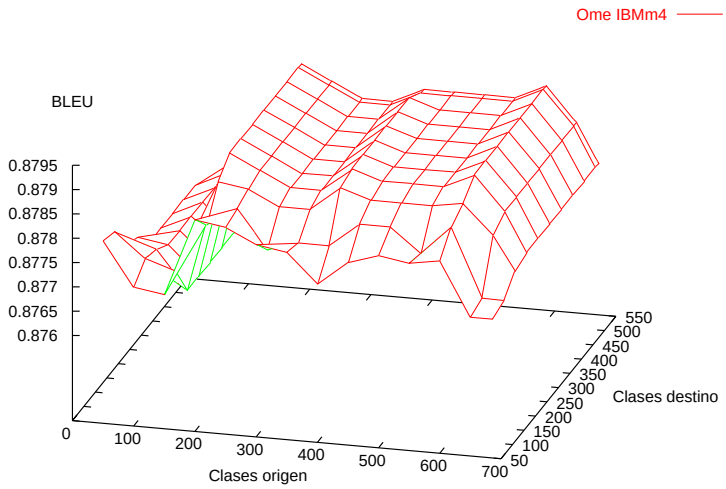
Como se puede observar en dichas figuras, la variación en el número de clases en la entrada y en la salida influye poco en los resultados finales de traducción. El BLEU varía entre 0,8761 y 0,8791 (diferencias en el tercer decimal) y el WER entre el 7 % y 7,12 %. Además, se puede observar en la figura 6.5 que apenas existe variación en los resultados obtenidos cuando en el lenguaje destino se utilizan entre 200 y 600 clases. Los mejores resultados (BLEU y WER) se alcanzan para el caso de 200 clases en el lenguaje origen y entre 200 y 513 clases para el lenguaje destino. A la vista de este experimento preliminar, el resto de alineamientos realizados sobre el corpus EuTrans I utilizando el modelo 4 de IBM han agrupado previamente los lenguaje origen y destino con 200 clases cada uno.

El segundo y último de los experimentos preliminares tiene por objeto determinar el parámetro b requerido por los algoritmos de agrupamiento *iterativo dejando uno fuera* e *incremental dejando uno fuera* (ver sección 3.5). En lugar de entrenar directamente dicho parámetro para cada experimento que utiliza dichos métodos, hemos preferido realizar un experimento preliminar para estudiar en qué grado afecta la variación de dicho parámetro a los resultados obtenidos y en el caso de que la variación no fuera grande, fijar un valor constante para dicho parámetro.

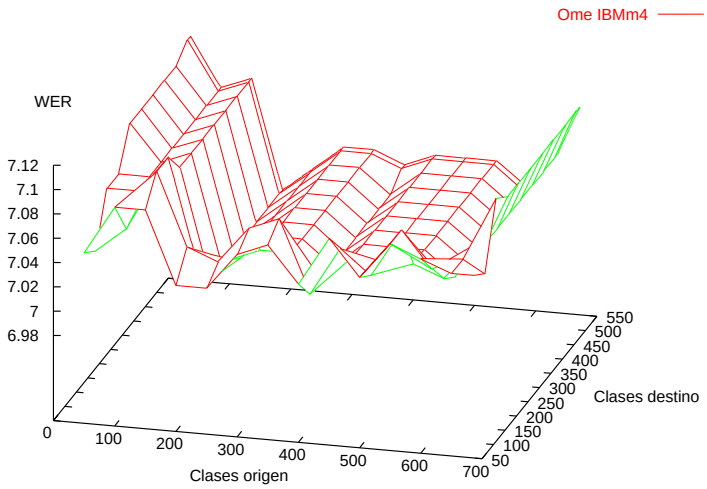
Para ello, hemos realizado una serie de experimentos de agrupamiento bilingüe utilizando como método de alineamiento el modelo 4 de IBM (con 200 clases en los lenguajes origen y destino). Como método de agrupamiento bilingüe hemos utilizado el método incremental dejando uno fuera. Los experimentos se han realizado para valores de b entre 0,1 y 0,95. Para cada valor de b se han realizado experimentos entre 100 y 500 clases bilingües. La figura 6.7 muestra los mejores BLEU y WER obtenidos para cada valor de b.

Como se puede apreciar, tanto el BLEU como el WER varían muy poco en función de b para el rango de 0,5 a 0,95 (el eje de ordenadas del BLEU va de 0,936 a 0,946 y el de WER de 3 a 3,7 %). Los mejores BLEU y WER, 0,9457 y 3,08 % se obtienen para $b = 0{,}9$ (con 375 clases).

A la vista de dicho experimento consideramos que se puede utilizar un único valor de b en los restantes experimentos debido a que los resultados obtenidos para distintos valores de b no presentarán grandes variaciones. De esta forma, evitamos tener que presentar por cada uno de los experimentos realizados, los resultados obtenidos para distintos valores de b, lo que clarifica la lectura e interpretación de resultados. Los restantes experimentos con el corpus EuTrans I

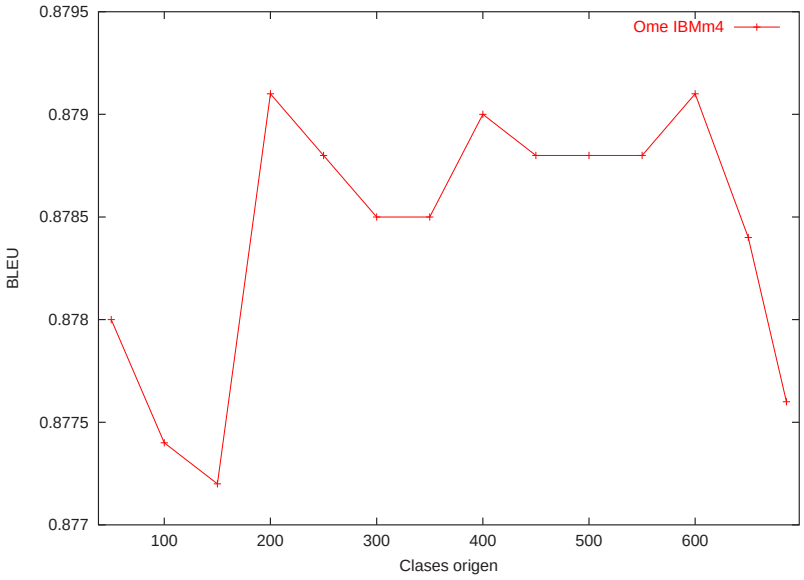


a)

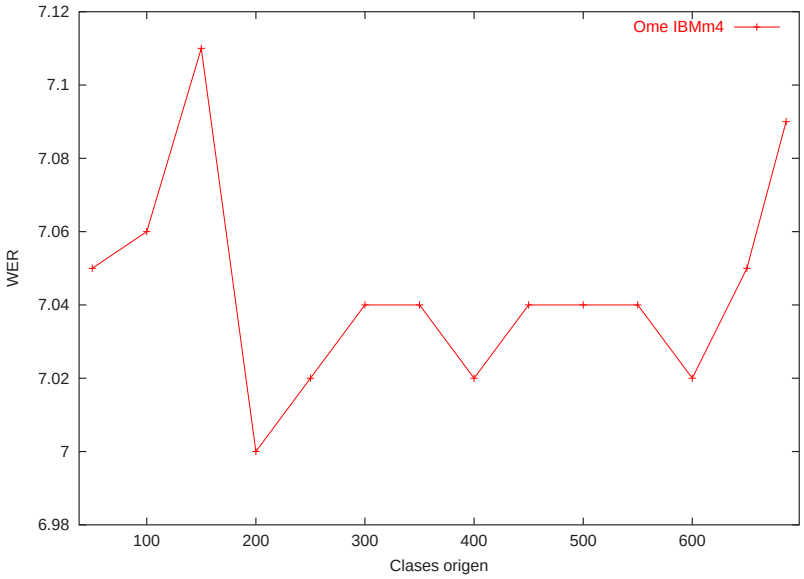


b)

Figura 6.5: a) BLEU y b) WER obtenidos con OMEGA utilizando el modelo 4 de IBM entrenado con distintos números de clases en los lenguaje origen y destino.

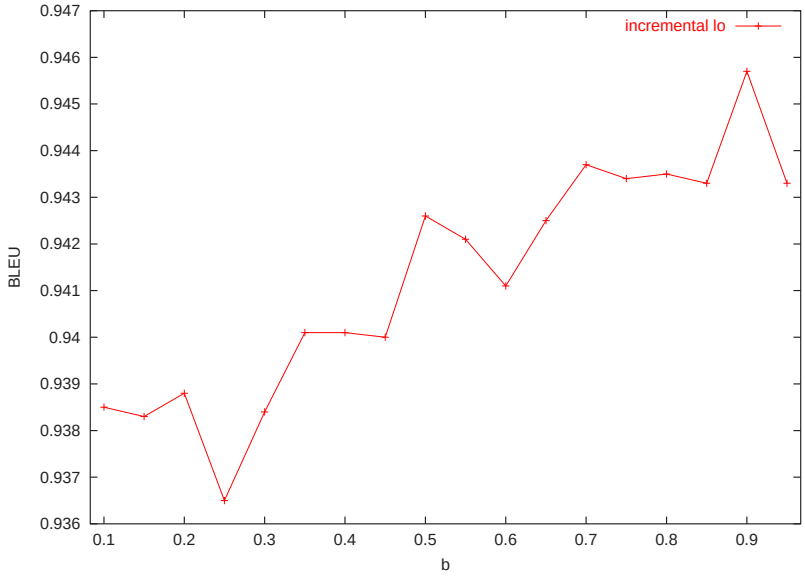


a)

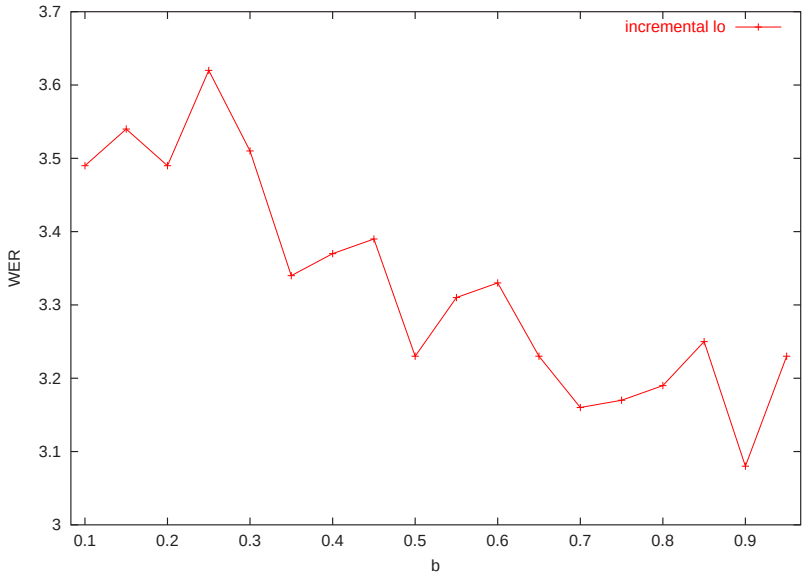


b)

Figura 6.6: Mejores a) BLEU y b) WER obtenidos con OMEGA utilizando el modelo 4 de IBM entrenado con distintos números de clases en el lenguaje origen. (✍ pág. 186)



a)



b)

Figura 6.7: a) BLEU y b) WER para distintos valores de b. (✍ pág. 186)

utilizarán $b = 0{,}9$.

Comparativa de distintos algoritmos de agrupamiento bilingüe

En esta subsección presentamos una serie de experimentos en la que se muestran los resultados obtenidos por distintos sistemas de traducción automática. Estos se han generado mediante el algoritmo OMEGA y el algoritmo OMEGA más clases.

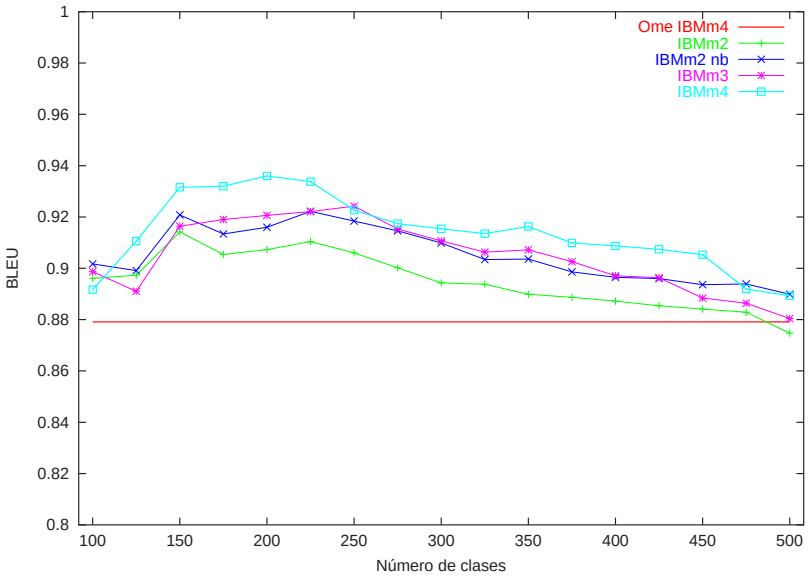
Los siguientes tres experimentos muestran la evolución del BLEU y del WER para traductores entrenados utilizando todas las frases de entrenamiento del corpus EuTrans I y distintos números de clases. Los algoritmos de agrupamiento bilingüe utilizados han sido el *iterativo* (experimento 1), el *iterativo dejando uno fuera* (experimento 2) y el *incremental dejando uno fuera* (experimento 3), respectivamente. Cada uno de los experimentos muestra, además, los diferentes resultados obtenidos en función del modelo de alineamiento utilizado tanto para la generación del agrupamiento bilingüe como para la inferencia del SST basado en clases.

Experimento 1. Con este experimento pretendemos comprobar si el el método *iterativo* de agrupamiento bilingüe mejora el aprendizaje de un sistema de traducción automática y la influencia del modelo de alineamiento escogido en dicha mejora. Para ello, hemos entrenado un sistema de traducción con OMEGA utilizando el modelo `IBMm4` y varios sistemas de traducción con OMEGA más clases para distintos números de clases y modelos de alineamiento.

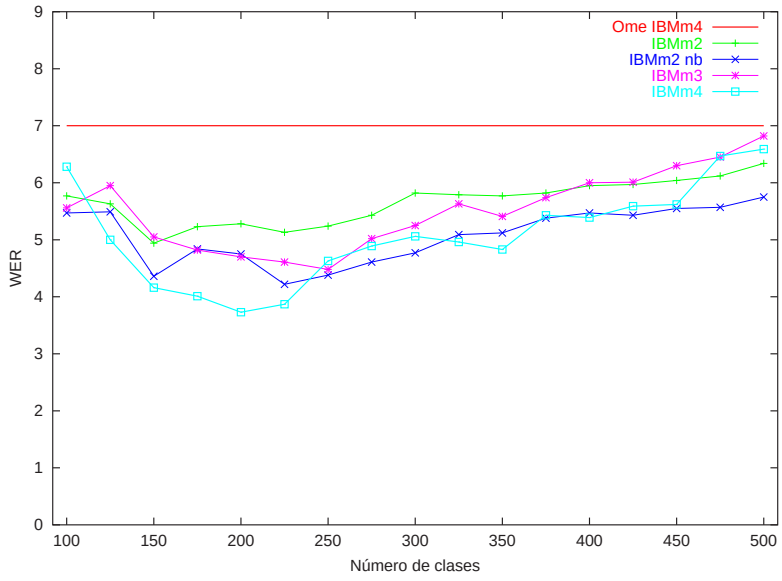
La figura 6.8 muestra el BLEU y el WER obtenidos con OMEGA utilizando el modelo 4 de IBM y la variación del WER y del BLEU con el número de clases obtenidos con OMEGA más las clases generadas por el algoritmo `iterativo`, para distintos modelos de alineamiento.

Como se puede observar, la evolución del BLEU con el número de clases es más estable que la del WER. Esto es debido a que el BLEU es una medida de la calidad de la traducción que tiene en cuenta más factores que el WER. No obstante, el comportamiento que se desprende de la interpretación a grandes rasgos de ambas medidas es el mismo. La traducción es mejor cuando se utilizan clases; tanto el BLEU como el WER son mejores en este caso. Los mejores resultados se obtienen entre 150 y 250 clases.

En cuanto al efecto del modelo de alineamiento utilizado, se puede observar lo siguiente. Si se comparan las gráficas correspondientes a los modelos `IBMm2` e `IBMm2_nb`, se puede observar como el segundo de ellos consigue mejores resultados. Además, la evolución de ambas gráficas es prácticamente paralela. Si se observa la evolución del BLEU se puede apreciar como los resultados obtenidos con el modelo `IBMm4` mejoran los de `IBMm3` y éstos a su vez están la mayor parte de las veces por encima de los obtenidos con el modelo `IBM2 nb`. En cuanto al WER, los resultados obtenidos por estos tres modelos presentan una serie de



a)



b)

Figura 6.8: a) BLEU y b) WER con OMEGA utilizando el modelo 4 de IBM y con OMEGA más las clases generadas por el algoritmo iterativo con números de clases entre 100 y 500 utilizando distintos modelos de alineamiento. (✍ pág. 187)

altibajos. Sin embargo, para el rango de clases óptimo, el modelo `IBMm4` supera también claramente a los anteriores.

Los mejores resultados de este experimento (método `iterativo`) se obtienen para 200 clases cuando se utiliza el modelo 4 de IBM y son: 0,9360 BLEU y 3,73 % WER. A efectos de comparación, los resultados de OMEGA con `IBMm4` sin utilizar clases son: 0,8791 BLEU y 7 % WER.

Experimento 2. Con este experimento pretendemos comprobar si el método *iterativo dejando uno fuera* mejora el aprendizaje de un sistema de traducción automática y la influencia del modelo de alineamiento escogido en dicha mejora. Para ello, hemos entrenado un sistema de traducción con OMEGA utilizando el modelo `IBMm4` y varios sistemas de traducción con OMEGA más clases para distintos números de clases y modelos de alineamiento.

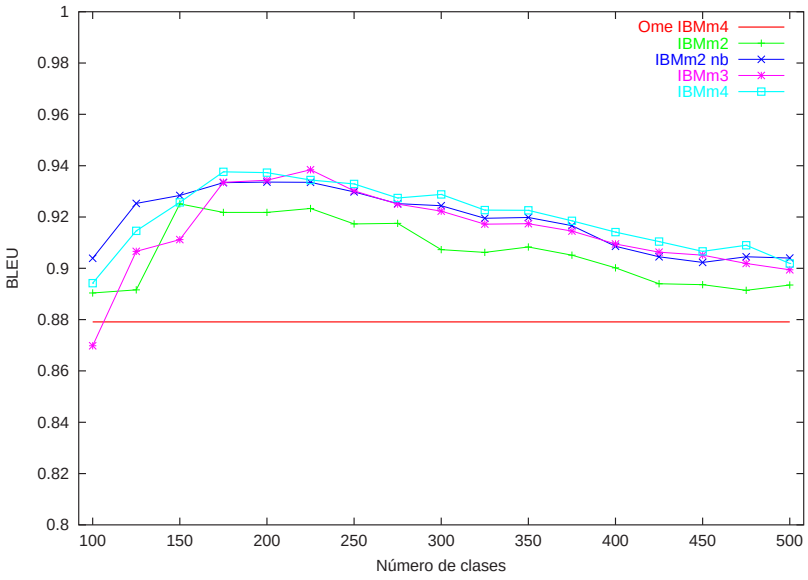
La figura 6.9 muestra el BLEU y el WER obtenidos con OMEGA utilizando el modelo 4 de IBM y la variación del WER y del BLEU con el número de clases obtenidos con OMEGA más las clases generadas por el algoritmo `iterativo_lo`, para distintos modelos de alineamiento.

Al igual que en el experimento anterior, la traducción es mejor cuando se utilizan clases; el BLEU y el WER son mejores en este caso. Los mejores resultados de este experimento se obtienen entre 150 y 300 clases. Si se compara esta gráfica y la correspondiente al método `iterativo` (figura 6.8) se puede apreciar como existe una mayor separación entre los resultados obtenidos con OMEGA y OMEGA más clases. Además, tanto el BLEU como el WER presentan un comportamiento más estable frente a la variación del número de clases que los obtenidos en el experimento anterior.

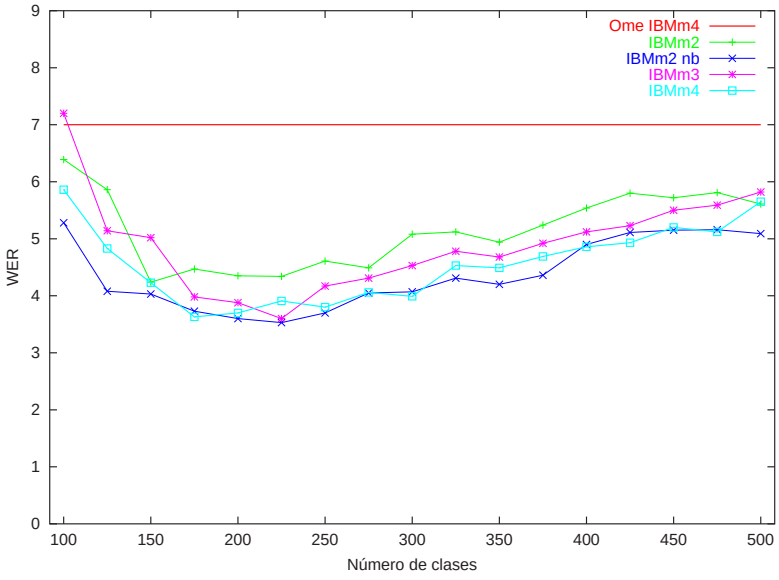
En cuanto al efecto del modelo de alineamiento utilizado, se puede observar lo siguiente. Si se comparan las gráficas correspondientes a los modelos `IBMm2` e `IBMm2_nb`, se puede observar como el segundo de ellos consigue mejores resultados. Además, al igual que en el experimento anterior, la evolución de ambas gráficas es prácticamente paralela. Si se observa la evolución del BLEU se puede apreciar como la diferencia entre los resultados obtenidos por los modelos `IBMm4`, `IBMm3` y `IBM2 nb` se ha reducido considerablemente, aunque por lo general el modelo `IBMm4` está ligeramente por encima de los otros dos en casi todos los casos. En cuanto al WER, curiosamente es el modelo `IBMm2_nb` el que obtiene los mejores resultados (a corta distancia de los modelos `IBMm4` y `IBMm3`).

Los mejores resultados de este experimento (método `iterativo_lo`) se obtienen para 220 clases cuando se utiliza el modelo 3 de IBM y son: 0,9384 BLEU y 3,6 % WER (OMEGA `IBMm4`: 0,8791 BLEU y 7 % WER). Para el modelo 4 los mejores resultados se obtienen para 175 clases y son: 0,9376 BLEU y 3,63 % WER, prácticamente los mismos que los obtenidos con el modelo 3.

Experimento 3. Con este experimento pretendemos comprobar si el método *incremental dejando uno fuera* mejora el aprendizaje de un sistema de traducción



a)



b)

Figura 6.9: a) BLEU y b) WER con OMEGA utilizando el modelo 4 de IBM y con OMEGA más las clases generadas por el algoritmo iterativo dejando uno fuera con números de clases entre 100 y 500 utilizando distintos modelos de alineamiento. (✍ pág. 189)

automática y la influencia del modelo de alineamiento escogido en dicha mejora. Para ello, hemos entrenado un sistema de traducción con OMEGA utilizando el modelo `IBMm4` y varios sistemas de traducción con OMEGA más clases para distintos números de clases y modelos de alineamiento.

La figura 6.10 muestra el BLEU y el WER obtenidos con OMEGA utilizando el modelo 4 de IBM y la variación del WER y del BLEU con el número de clases obtenidos con OMEGA más las clases generadas por el algoritmo `incremental_lo`, para distintos modelos de alineamiento.

Al igual que en los experimentos anteriores, la traducción es mejor cuando se utilizan clases; el BLEU y el WER son mejores en este caso. Si se comparan los resultados obtenidos con los correspondientes al método *iterativo dejando uno fuera* (figura 6.9) se puede observar como la curva de los resultados obtenidos con respecto al número de clases es aún más plana. Una consecuencia de que los resultados sean tan similares para un gran rango de clases es que para los distintos métodos de alineamiento utilizados se obtienen los mejores resultados en distintos rangos de clases.

En cuanto al efecto del modelo de alineamiento utilizado, se puede observar lo siguiente. Los mejores resultados se obtienen con el modelo `IBMm4`, seguidos de los obtenidos con el modelo `IBMm3`, luego los del modelo `IBMm2_nb` y por último los del modelo `IBMm2`.

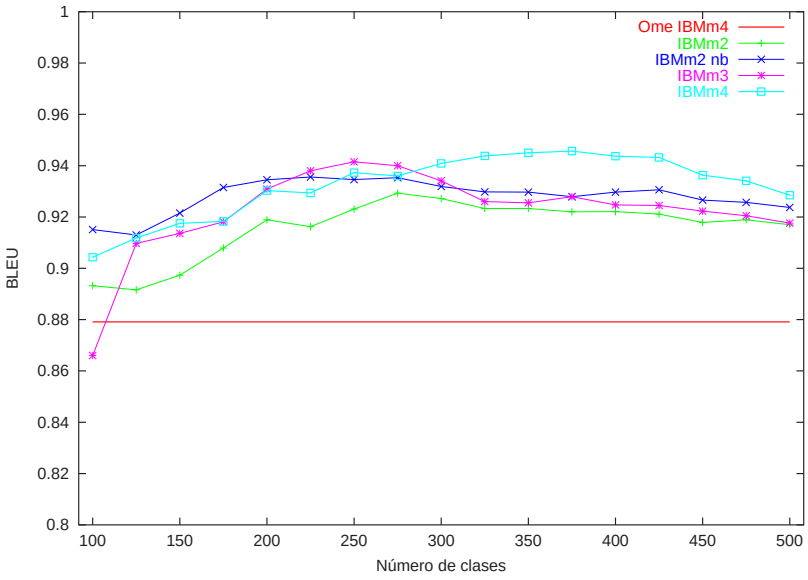
Los mejores resultados de este experimento (método `incremental_lo`) se obtienen para 375 clases cuando se utiliza el modelo 4 de IBM y son: 0,9457 BLEU y 3,08 % WER (OMEGA `IBMm4`: 0,8791 BLEU y 7 % WER).

Conclusiones experimentos 1–3. Los tres experimentos anteriores muestran que los resultados de traducción obtenidos para el corpus EuTrans I (con 10.000 frases de entrenamiento) son mejores cuando se utiliza cualquiera de los métodos de agrupamiento bilingüe propuestos que cuando no. El método `incremental_lo` obtiene mejores resultados que el `iterativo_lo`, y éste a su vez mejores que el `iterativo`. Además, el método `incremental_lo` es más estable con el número de clases que el método `iterativo_lo`, y éste a su vez es más estable que el método `iterativo`.

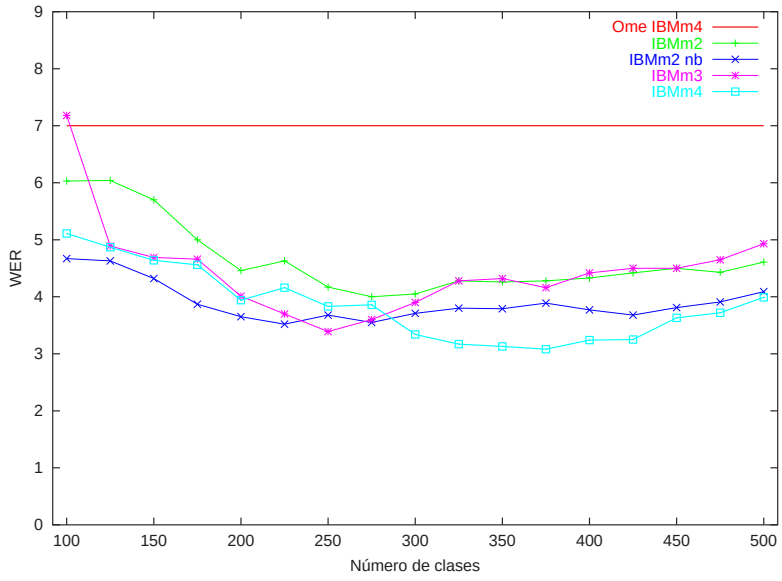
En cuanto a los modelos de alineamiento, los resultados obtenidos muestran lo que cabía esperar, que cuanto mejor es el método de alineamiento utilizado, mejores serán tanto los agrupamientos como los traductores automáticos inferidos.

El cuadro 6.9 resume los mejores BLEU y WER obtenidos con OMEGA y con OMEGA más clases para los distintos métodos de agrupamiento bilingüe; todos ellos utilizando el modelo 4 de IBM.

Experimento 4. Con este experimento pretendemos estudiar cómo afecta el número de frases utilizadas para el entrenamiento a la inferencia de los sistemas de traducción basados en agrupamientos bilingües. También queremos observar si



a)



b)

Figura 6.10: a) BLEU y b) WER con OMEGA utilizando el modelo 4 de IBM y con OMEGA más las clases generadas por el algoritmo incremental dejando uno fuera con números de clases entre 100 y 500 utilizando distintos modelos de alineamiento. (✍ pág. 192)

Método	BLEU	WER
OMEGA (Ω)	0,8791	7
Ω + `iterativo`	0,9360	3,73
Ω + `iterativo_lo`	0,9376	3,63
Ω + `incremental_lo`	0,9457	3,08

Cuadro 6.9: BLEU y WER con OMEGA y mejores BLEU y WER con OMEGA con clases para distintos métodos de agrupamiento bilingüe utilizando IBMm4.

la relación entre los métodos de agrupamiento bilingüe se mantiene con distintos números de frases de entrenamiento. Para ello, hemos dividido el corpus de entrenamiento en particiones de 1.000, 2.000, ... ,10.000 frases. Para cada una de estas particiones hemos entrenado un sistema de traducción con OMEGA y tres sistemas de traducción con OMEGA más clases (cada uno con uno de los métodos de agrupamiento bilingüe). Para cada sistema de traducción OMEGA más clases hemos hecho pruebas con distintos números de clases entre 100 y 500 y nos hemos quedado con aquel número de clases con el que se obtiene el mejor BLEU. Como modelo de alineamiento hemos utilizado en todos los experimentos el `IBMm4`. La figura 6.11 representa los resultados obtenidos.

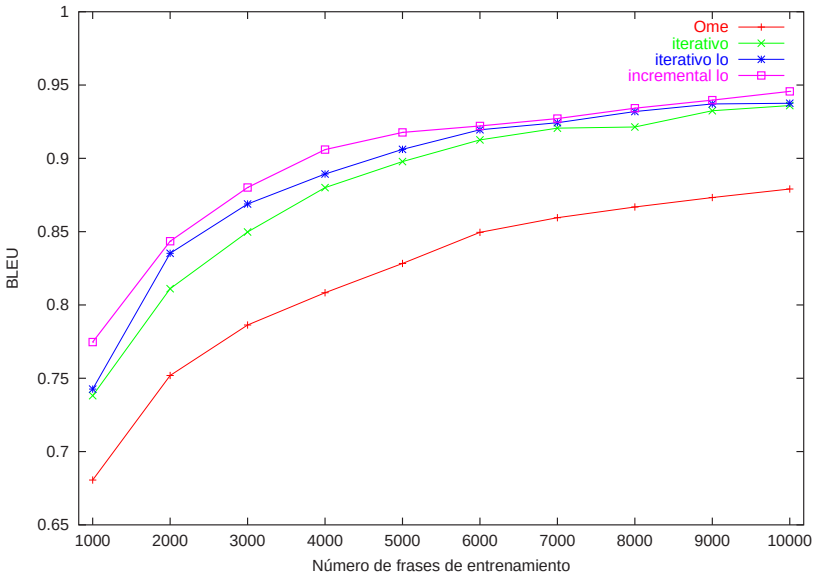
Como se puede observar, la traducción es mejor cuando se utilizan clases para todos los conjuntos de entrenamiento probados. Un aspecto que podríamos destacar, a la vista de los resultados de este experimento, es que con sólo 3.000 frases de entrenamiento obtenemos mejores resultados con clases que con 10.000 cuando no se utilizan.

En cuanto a la comparación entre los distintos métodos de agrupamiento bilingüe, al igual que en los experimentos 1–3, se puede apreciar que el método `incremental_lo` produce mejores resultados que el método `iterativo_lo`, y éste último mejores que el `iterativo` para cualquier número de frases de entrenamiento.

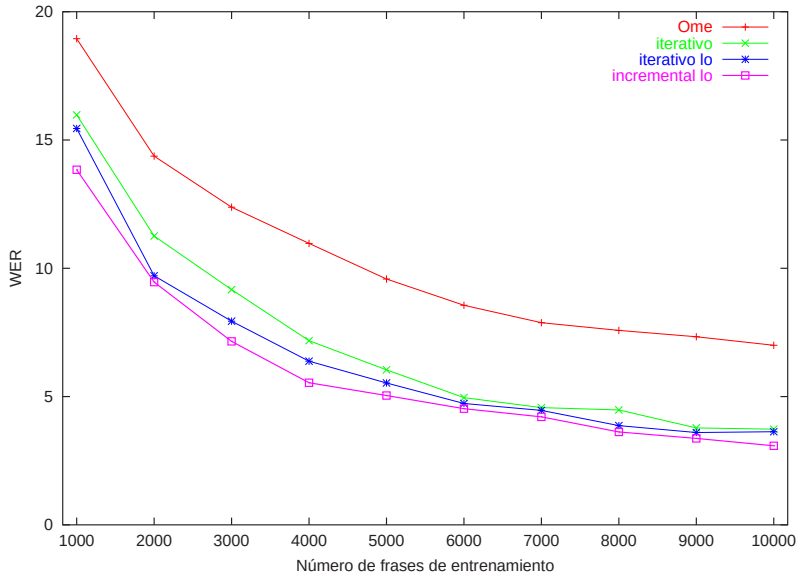
Estimación del número óptimo de clases

Como se comentó en las secciones 3.5 y 3.6, tanto el método `iterativo_lo` como el método `incremental_lo` utilizan la técnica de dejar uno fuera para simular eventos no vistos e intentar evitar el sobreentrenamiento. Como vimos en la sección 6.2, la variación de la función de optimización de estos métodos con el número de clases presenta un mínimo. Suponemos que el número de clases en el que se alcanza dicho mínimo corresponde al número óptimo de clases.

Experimento 5. Con este experimento pretendemos verificar cómo afecta al sistema de traducción automática el hecho de que el agrupamiento se realice con un determinado número de clases calculado de forma automática. Para ello, hemos



a)



b)

Figura 6.11: a) BLEU y b) WER con OMEGA y con OMEGA más las clases generadas por distintos métodos de agrupamiento bilingüe entrenados con distintos números de frases entre 1.000 y 10.000. (pág. 195)

entrenado un sistema de traducción con OMEGA más las clases generadas con el método `incremental_lo` para cada partición del conjunto de entrenamiento entre 1.000 y 10.000 pares de frases. En lugar de probar distintos números de clases, se ha dejado que el método determine el número óptimo de clases. Compararemos estos resultados con los valores ya obtenidos en el experimento anterior.

La figura 6.12 muestra el BLEU y el WER obtenidos con OMEGA, con OMEGA más clases con el mejor número de clases y con OMEGA más clases con el número óptimo determinado de forma automática. Para obtener el mejor número de clases se han realizado pruebas con distintos números de clases entre 100 y 500 y nos hemos quedado con el número de clases con el que se obtiene el mejor BLEU (igual que en el experimento 4). El método de agrupamiento ha sido el `incremental_lo`.

Como se puede observar en dicha figura, los resultados obtenidos para el número de clases óptimo determinado por el entrenamiento son prácticamente iguales a obtenidos para el mejor número de clases, especialmente hasta las 6.000 frases de entrenamiento. Para 7.000 a 10.000 frases de entrenamiento se obtienen resultados ligeramente mejores para el mejor número de clases que para el determinado como óptimo por el entrenamiento.

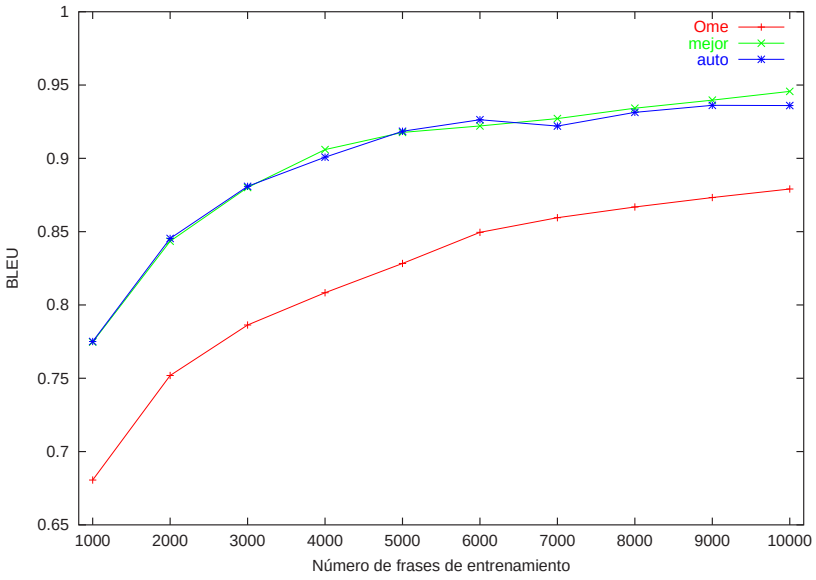
Podríamos concluir, a la vista de este experimento, que el número determinado como óptimo según el agrupamiento puede utilizarse satisfactoriamente para entrenar de forma automática un traductor basado en dicho número de clases.

Identificación de colocaciones

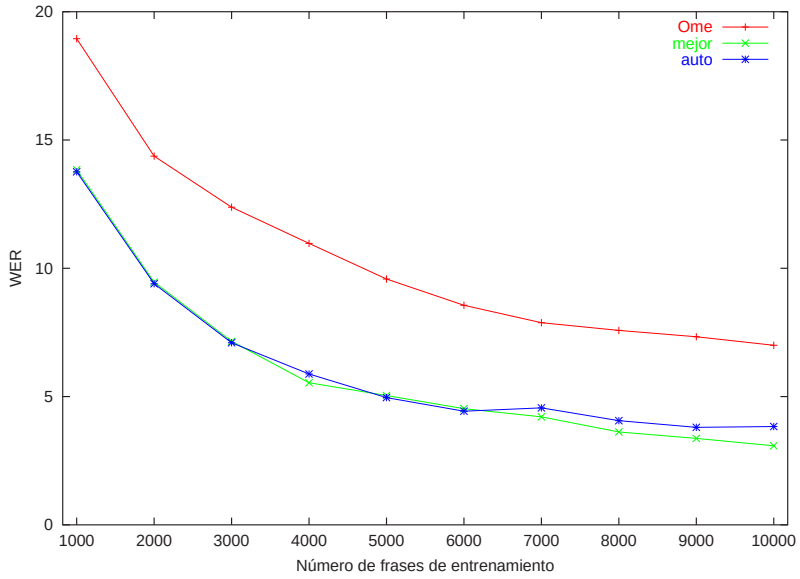
Los siguientes experimentos tienen por objeto comprobar cómo afecta la identificación automática de colocaciones a los sistemas de traducción generados.

Experimento 6. Este experimento pretende comprobar si la identificación automática de colocaciones permite mejorar el entrenamiento de sistemas de traducción. Para ello, se han identificado las colocaciones para cada uno de las particiones de 1.000 a 10.000 frases de entrenamiento del corpus EuTrans I mediante el algoritmo descrito en el capítulo 5 (utilizando el modelo 4 de IBM). A continuación, se han generado los agrupamientos bilingües con distintos números de clases entre 100 y 500 de cada uno de los corpora en los que previamente se han marcado las colocaciones. Por último, con estos agrupamientos con colocaciones se han inferido los correspondientes SSTs utilizando OMEGA.

La figura 6.13 muestra los resultados de este experimento. En dicha figura se muestran el BLEU y el WER obtenidos con OMEGA (ver experimento 4), con OMEGA con clases (ver experimento 4) y con OMEGA con clases y colocaciones. También se muestra los resultados obtenidos con OMEGA entrenado sobre los corpora con colocaciones. El algoritmo de agrupamiento bilingüe utilizado ha sido el `incremental_lo`.



a)



b)

Figura 6.12: a) BLEU y b) WER con OMEGA, OMEGA más clases para el mejor número de clases (`mejor`) y OMEGA más clases con el número óptimo determinado por el entrenamiento (`auto`) entrenados con distintos números de frases entre 1.000 y 10.000. (✍ pág. 196)

Puesto que la identificación de las colocaciones tiene como objeto la obtención de mejores agrupamientos bilingües, el uso de colocaciones con OMEGA, como era de esperar, no presenta una gran variación en la calidad de las traducciones generadas. Donde la identificación de colocaciones sí que mejora los resultados, tanto de BLEU como de WER, es cuando se utiliza conjuntamente con el agrupamiento bilingüe para la inferencia de los traductores. De hecho, con 7.000 frases de entrenamiento se alcanzan los mismos resultados utilizando clases más colocaciones que los que se consiguen con 10.000 frases de entrenamiento cuando se utilizan clases pero no colocaciones.

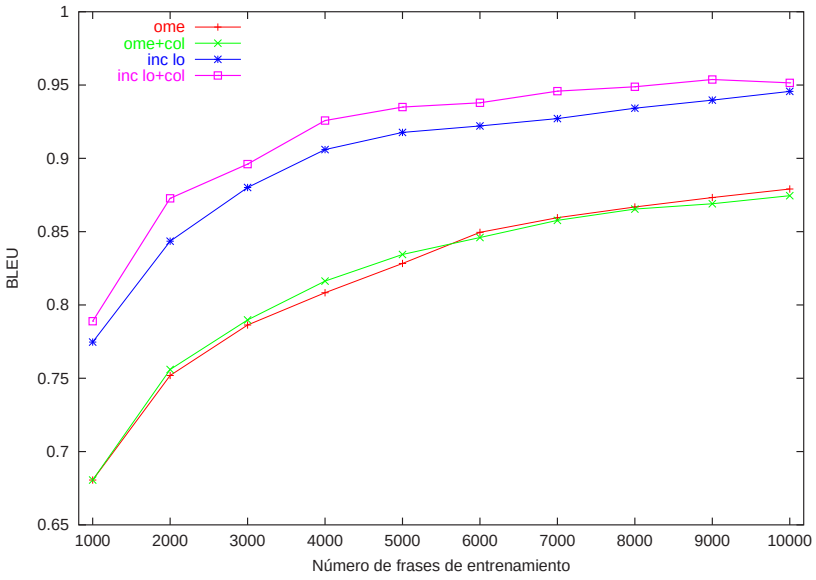
Experimento 7. Con este experimento pretendemos estudiar cómo afecta la identificación de colocaciones en la mejora de sistemas de traducción cuando variamos el número de clases. Para ello, hemos realizado un experimento en el que hemos identificado de forma automática las colocaciones en el corpus EuTrans I con 10.000 frases de entrenamiento y hemos entrenado traductores con OMEGA más clases y OMEGA más clases y colocaciones con distintos números de clases entre 100 y 500.

La figura 6.14 muestra los resultados obtenidos con distintos números de clases entre 100 y 500 de OMEGA, con OMEGA con clases bilingües utilizando el método `incremental_lo` sin colocaciones (ver figura 6.11) y con clases bilingües y colocaciones. Como se puede observar, para cualquier número de clases de los mostrados, los resultados obtenidos son mejores cuando se utilizan colocaciones.

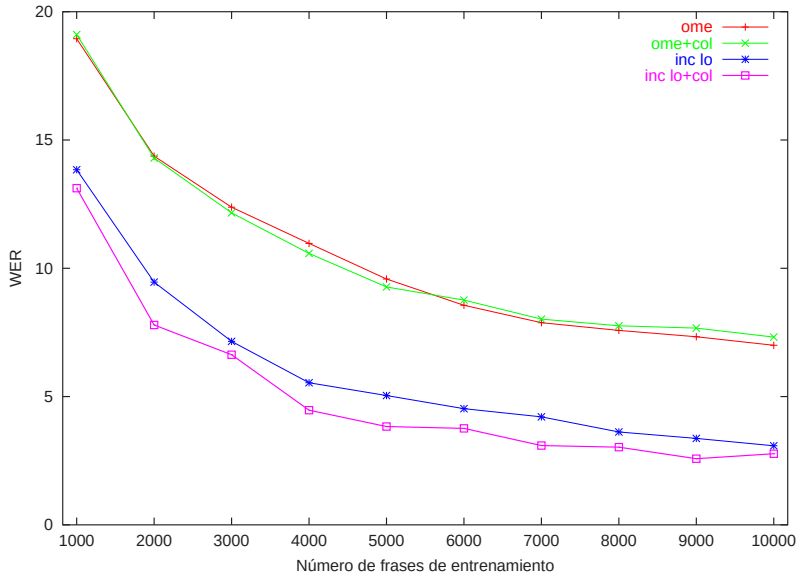
Experimento 8. Con este experimento queremos comprobar si cuando utilizamos colocaciones y determinamos de forma automática el número óptimo de clases obtenemos traducciones de calidad similar a las obtenidas para el mejor número de clases. Para ello, hemos identificado de forma automática las colocaciones sobre las particiones de 1.000 a 10.000 frases del corpus EuTrans I y hemos generado sistemas de traducción con OMEGA más clases y colocaciones sobre estos corpora. Hemos dejado que el método de agrupamiento bilingüe determine de forma automática el número óptimo de clases. Hemos comparado estos resultados con los mostrados en el experimento 6 obtenidos con OMEGA más clases y colocaciones para el mejor número de clases.

La figura 6.15 muestra los mejores resultados obtenidos con 1.000 a 10.000 frases de entrenamiento utilizando distintos números de clases entre 100 y 500 y los resultados obtenidos utilizando el número de clases estimado como óptimo. Como se puede observar, las gráficas son prácticamente idénticas, por lo que podemos considerar válida la estimación del número de clases óptimo.

Experimento 9. Este experimento pretende comparar la calidad de los sistemas de traducción OMEGA más clases y OMEGA más clases y colocaciones cuando el número de clases se determina de forma automática en ambos casos. Para ello, se comparan los resultados obtenidos en el experimento 5 con los traductores

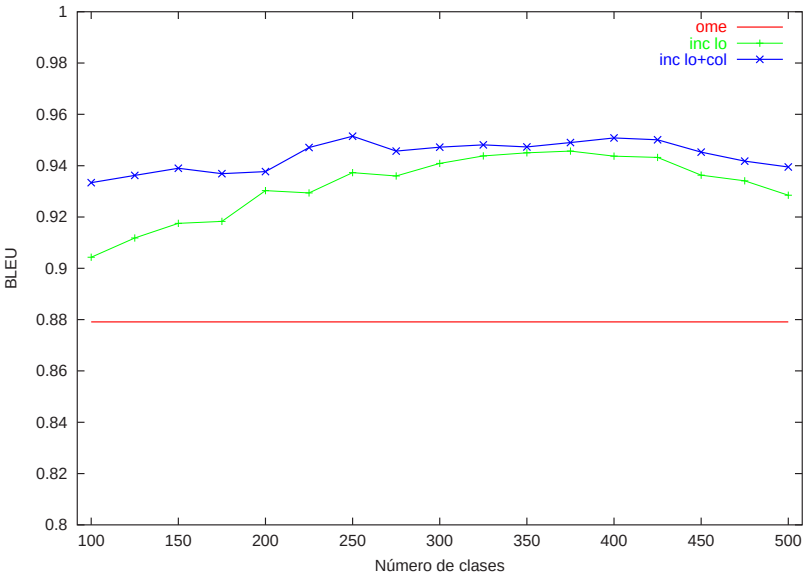


a)

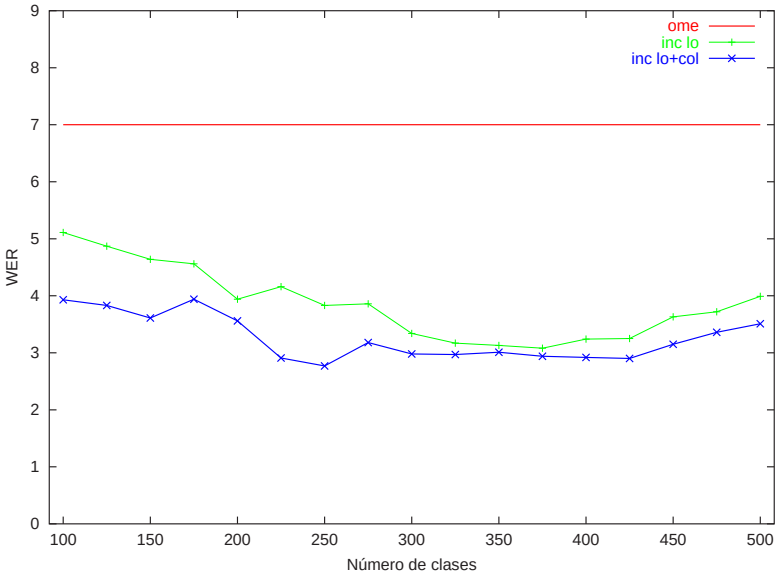


b)

Figura 6.13: a) BLEU y b) WER con OMEGA sin colocaciones (`ome`), OMEGA con colocaciones (`ome+col`), y los mejores resultados con OMEGA más clases (`inc lo`) y con OMEGA más clases y colocaciones (`inc lo+col`) entrenados con distintos números de frases entre 1.000 y 10.000. (pág. 197)

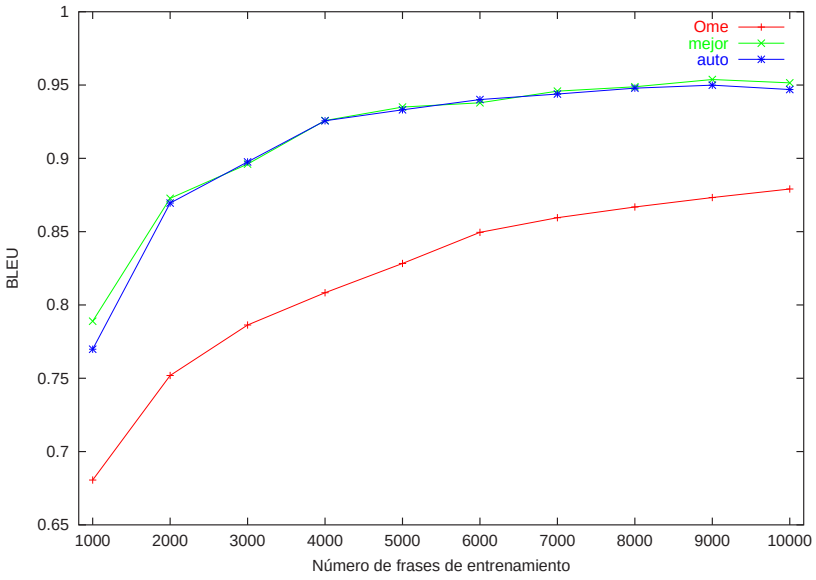


a)

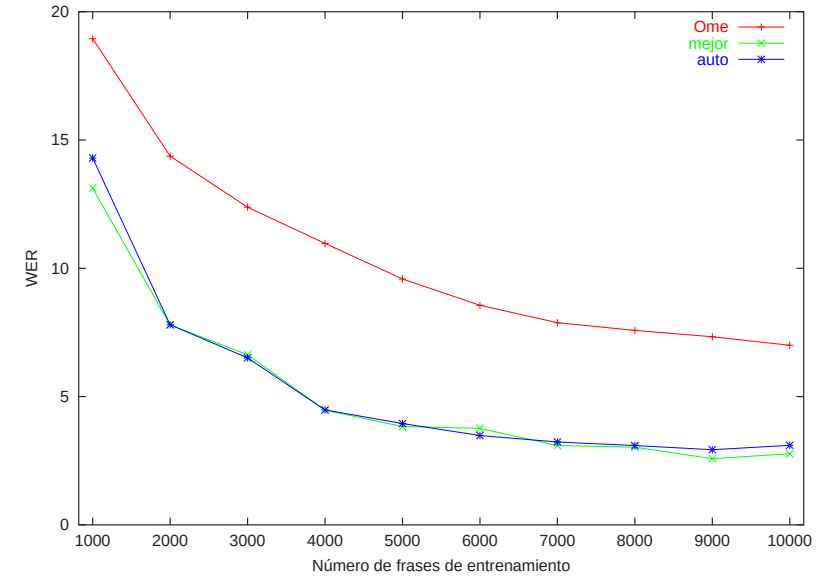


b)

Figura 6.14: a) BLEU y b) WER con OMEGA sin colocaciones (`ome`), OMEGA más clases (`inc lo`) y OMEGA más clases y colocaciones (`inc lo+col`) con distintos números de clases entre 100 y 500 entrenados con 10.000 frases de entrenamiento. (pág. 197)



a)



b)

Figura 6.15: a) BLEU y b) WER con OMEGA, con OMEGA más clases y colocaciones para el mejor número de clases (`mejor`) y OMEGA más clases y colocaciones con el número óptimo determinado por el entrenamiento (`auto`) entrenados con entre 1.000 y 10.000 frases de entrenamiento. (pág. 198)

inferidos con entre 1.000 y 10.000 frases de entrenamiento con OMEGA más clases y colocaciones donde el número óptimo de clases se ha determinado de forma automática.

La figura 6.16 muestra el BLEU y WER obtenido con los distintos números de frases de entrenamiento con OMEGA, con OMEGA más clases (ver figura 6.12) y con OMEGA más clases y colaciones. Estos dos últimos métodos utilizan el número óptimo de clases determinado por el método de agrupamiento bilingüe `incremental_lo`. Como se puede observar, los mejores resultados se obtienen, nuevamente, cuando se utilizan colocaciones.

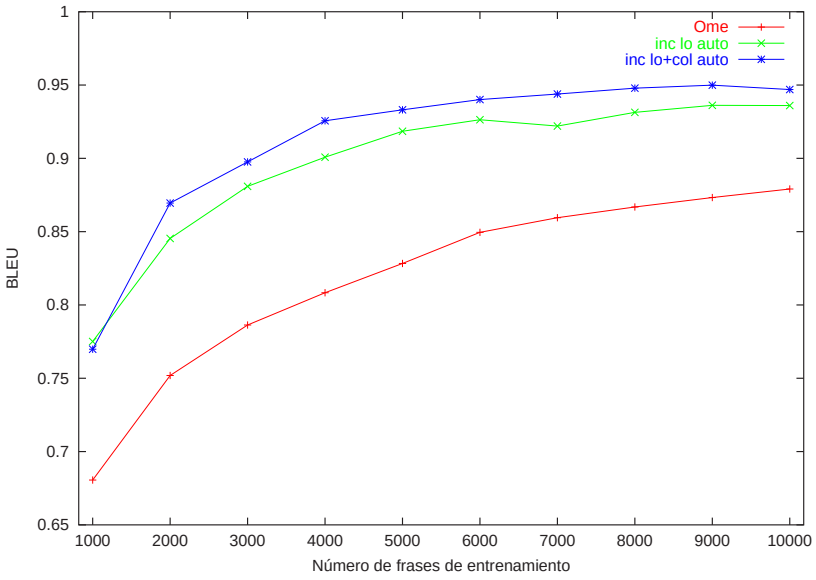
Más resultados de colocaciones. En la sección A.3 se muestran las colocaciones identificadas de forma automática sobre el corpus EuTrans I con 10.000 frases. Y en la sección A.4 se muestra el agrupamiento obtenido utilizando estas colocaciones. Este agrupamiento puede compararse con el mostrado en la sección A.2 para el que no se han utilizado colocaciones.

Comparación con resultados previos

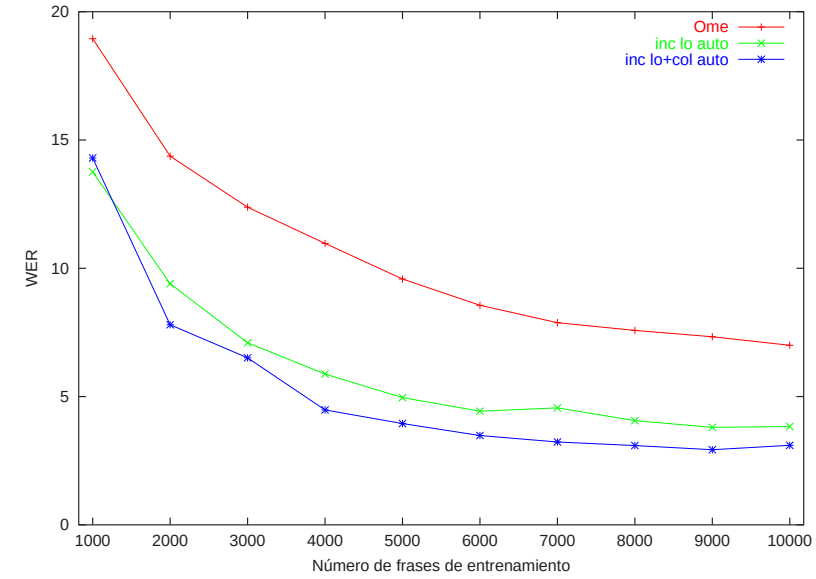
El cuadro 6.10 muestra el WER obtenido sobre el corpus EuTrans I con 10.000 muestras de entrenamiento para distintos sistemas de traducción. El primero de los valores mostrados corresponde al WER obtenido utilizando únicamente el algoritmo OMEGA, que es 7,0 %. El segundo de los valores, 3,0 %, es el WER obtenido utilizando un conjunto de categorías bilingües definidas a mano tal y como se describe en [ABC+97a] (ver sección 2.6). Los siguientes dos resultados, OMEGA+CB y OMEGA+PE, corresponden a los mejores resultados obtenidos utilizando `IBMm2_nb` con clases bilingües y con el método de identificación de compuestos mediante estimación parcial (*Partial Estimation* en [IFRZ00]), respectivamente. Este último corresponde al mejor resultado obtenido durante el proyecto EuTrans (proyecto ESPRIT n. 30268) de entre los sistemas de traducción automática desarrollados en dicho proyecto para el corpus EuTrans I (sin considerar las categorías manuales).

El método OMEGA+CB, propuesto previamente, se ha mejorado en esta tesis dando lugar al método de agrupamiento bilingüe `iterativo` y al nuevo método de integración con OMEGA. El método de detección de compuestos presentado en [BV01], propuesto por el autor, y que da lugar al método OMEGA+PE [IFRZ00], se ha mejorado en esta tesis dando lugar al método propuesto de identificación de colocaciones.

El siguiente resultado en el cuadro 6.10, OMEGA+iter (`IBMm2_nb`) muestra como el resultado obtenido por el nuevo método `iterativo` supera al obtenido con OMEGA+CB utilizando el mismo modelo de alineamiento. Además, utilizando el método `incremental_lo` con `IBMm2_nb`, sin colocaciones, obtenemos mejores resultados que cuando además de las clases bilingües utilizábamos compuestos (OMEGA+PE).



a)



b)

Figura 6.16: a) BLEU y b) WER con OMEGA, con OMEGA más clases (`inc lo`) y con OMEGA más clases y colocaciones (`inc lo+col`) con el número óptimo determinado por el entrenamiento entrenados con distintos números de frases entre 1.000 y 10.000.

Método	WER	Referencia
OMEGA	7,0	[IFRZ00]
OMEGA+CBM	3,0	[IFRZ00]
OMEGA+CB	5,19	[BV99c]
OMEGA+PE	3,9	[IFRZ00]
OMEGA+iter (`IBMm2_nb`)	4,22	
OMEGA+inc lo (`IBMm2_nb`)	3,52	
OMEGA+iter (`IBMm4`)	3,73	
OMEGA+inc lo (`IBMm4`)	3,08	
OMEGA+inc lo+col (`IBMm4`)	2,77	

Cuadro 6.10: Comparativa de distintos sistemas de traducción automática con EuTrans I.

Los tres últimos métodos presentados, OMEGA+iter, OMEGA+inc lo, OMEGA+inc lo+col, corresponden a los métodos `iterativo`, `incremental_lo` e `incremental_lo` más colocaciones utilizando el modelo `IBMm4`. Como ya hemos visto anteriormente, el uso de un mejor modelo de alineamiento conlleva mejores resultados. Además, ahora el método `iterativo` también supera al resultado anterior obtenido con el método OMEGA+PE.

Por último, compararamos el WER obtenido anteriormente con categorías bilingües manuales (OMEGA+CBM) y el obtenido con el agrupamiento bilingüe con el método `incremental_lo` y colocaciones (OMEGA+inc lo+col). Este último método obtiene un valor, 2,77 %, inferior al publicado con categorías manuales, 3,0 %. Esto nos anima a creer que las técnicas propuestas en esta tesis son adecuadas.

Esta última comparación no es del todo justa ya que el resultado de OMEGA+CBM se obtuvo en su momento utilizando el modelo `IBMm2`. Se podría rehacer dicho experimento con el modelo `IBMm4` para comparar el nuevo resultado con el 2,77 % obtenido de forma automática. Sin embargo, aún en el caso de que con categorías manuales se consiguiera un menor WER, la aproximación automática seguiría siendo buena ya que evita recurrir a conocimiento experto y la diferencia entre ambos no sería en este caso muy grande (ya que el WER actual es muy bajo).

6.3.2. Experimentos con EuTrans II

Los experimentos presentados sobre el corpus EuTrans II utilizan el modelo de alineamiento `IBMm4` y el algoritmo de agrupamiento bilingüe `incremental_lo`. Se han utilizado estos métodos, por ser los que mejores resultados han obtenido en los experimentos realizados con EuTrans I.

Experimentos preliminares

Como ya se expuso en la sección 6.3.1, el modelo 4 de IBM requiere el agrupamiento monolingüe de los lenguajes origen y destino para la estimación de sus parámetros.

El número de clases utilizado en los restantes experimentos para el agrupamiento de los lenguajes origen y destino se ha determinado realizando el siguiente experimento preliminar. Para cada posible par de número de clases se ha entrenado un SST con el alineamiento generado utilizando dichos agrupamientos. Este SST se ha utilizado para traducir el test. Por último, se han seleccionado los números de clases del lenguaje origen y del lenguaje destino que han proporcionado la mejor traducción. Al proceder de esta forma, las técnicas de agrupamiento bilingüe propuestas en esta tesis utilizarán el mismo alineamiento que el que conduce a la mejor traducción posible cuando no se utilizan.

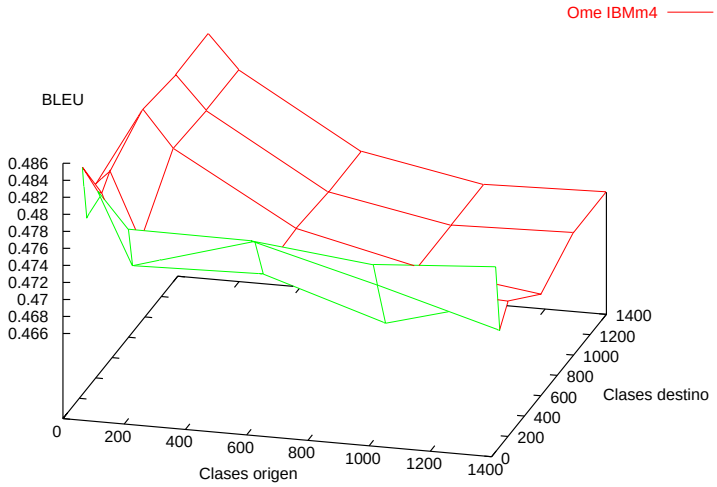
La figura 6.17 muestra la variación de BLEU y WER con el número de clases de entrada y salida. La figura 6.18 muestra la variación del mejor BLEU y WER con el número de clases de entrada.

Como se puede observar en dichas figuras, la variación en el número de clases en la entrada y en la salida influye poco en los resultados finales de traducción. El BLEU varía entre 0,4767 y 0,4855 y el WER entre el 35,79 % y 36,50 %. Los mejores resultados (BLEU y WER) se alcanzan para el caso de 100 clases en el lenguaje origen y 600 clases en el lenguaje destino. A la vista de este experimento preliminar, el resto de alineamientos realizados sobre el corpus EuTrans II utilizando el modelo 4 de IBM han agrupado previamente los lenguaje origen y destino con 100 y 600 clases, respectivamente.

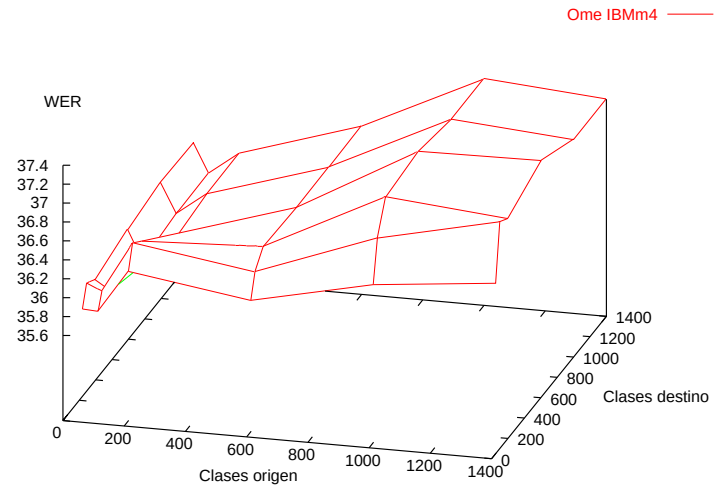
Experimentos de agrupamiento bilingüe

Experimento 1. Este experimento pretende comprobar si el agrupamiento bilingüe puede mejorar el aprendizaje de sistemas de traducción sobre el corpus EuTrans II. Para ello, se ha entrenado traductores con OMEGA y con OMEGA más las clases generadas mediante el algoritmo `incremental_lo` con distintos números de clases entre 200 y 2.869. Donde 2.869 es el número máximo de clases (tantas clases como palabras extendidas). Como el algoritmo `incremental_lo` depende de un parámetro b (ver sección 3.5) se han realizado agrupamientos con b igual a 0,5, 0,7 y 0,9.

La figura 6.19 muestra el BLEU y el WER obtenidos para este experimento. Como se puede observar, los resultados obtenidos utilizando clases no mejoran a los obtenidos sin clases independientemente del valor de b. De hecho, cuanto mayor es el número de clases más cercanos se encuentran tanto el BLEU como el WER a los obtenidos sin clases. Este resultado era predecible a la vista de los experimentos de agrupamiento monolingüe realizados en la sección 6.2.2 sobre el corpus EuTrans II en los que ya se adelantaba que los agrupamientos generados sobre dicho corpus no eran buenos. Suponemos que esto es debido al escaso

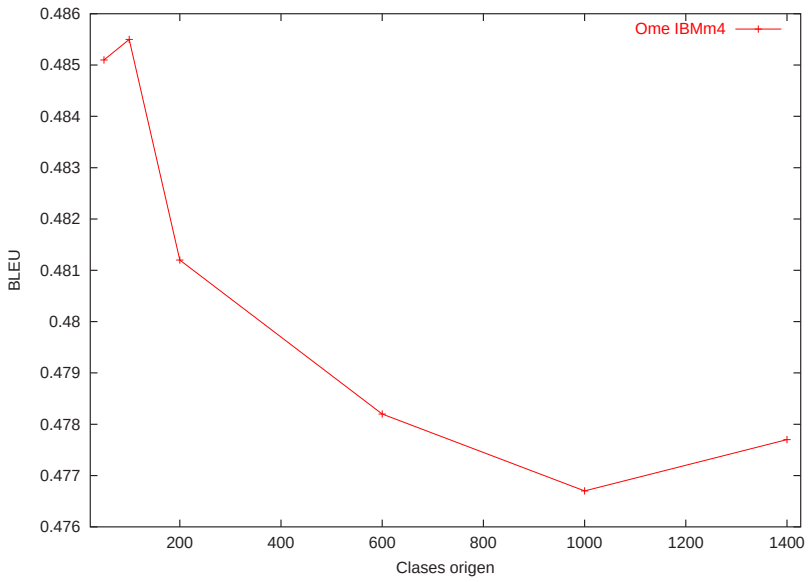


a)

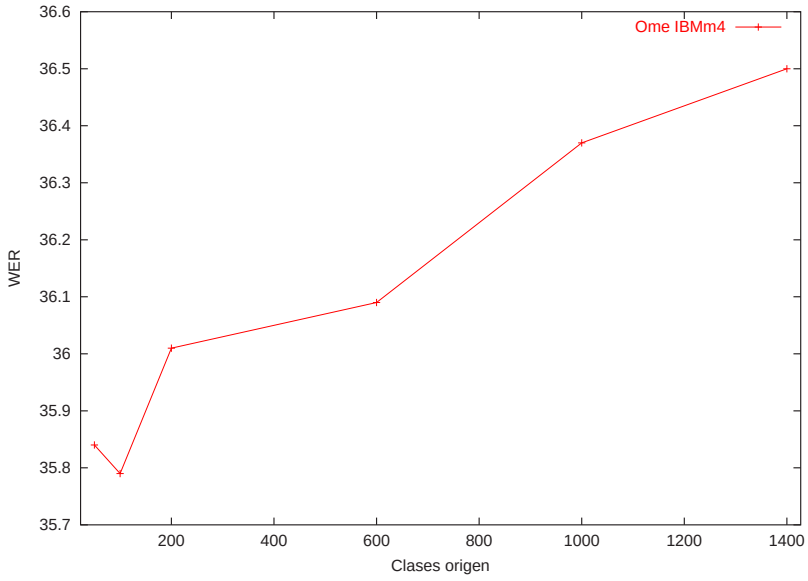


b)

Figura 6.17: a) BLEU y b) WER con OMEGA utilizando el modelo 4 de IBM entrenado con distintos números de clases en los lenguaje origen y destino.



a)



b)

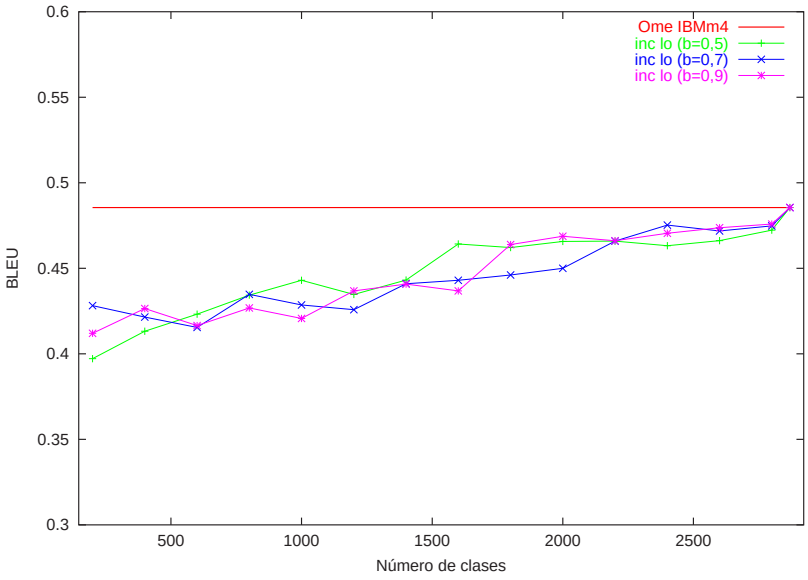
Figura 6.18: Mejores a) BLEU y b) WER con OMEGA utilizando el modelo 4 de IBM entrenado con distintos números de clases en el lenguaje origen. (✍ pág. 198)

número de frases de entrenamiento.

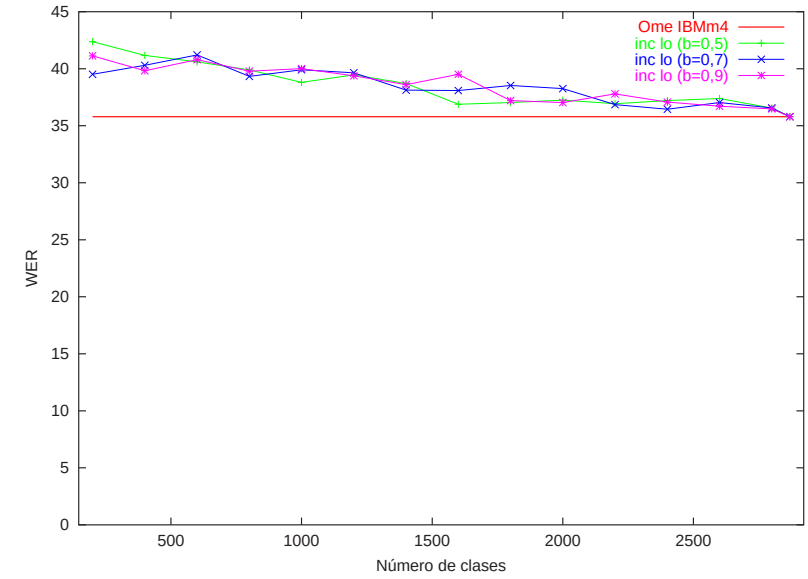
Experimentos de agrupamiento bilingüe y colocaciones

Experimento 2. Este experimento pretende comprobar si el agrupamiento bilingüe con colocaciones puede mejorar el aprendizaje de sistemas de traducción sobre el corpus EuTrans II. Para ello, se han identificado de forma automática las colocaciones sobre el corpus EuTrans II. Utilizando este corpus con colocaciones se ha entrenado traductores con OMEGA y con OMEGA más las clases generadas mediante el algoritmo `incremental_lo` para distintos números de clases entre 200 y 3.511 clases. Donde 3.511 es el número máximo de clases (tantas clases como palabras extendidas con colocaciones). Como el algoritmo `incremental_lo` depende de un parámetro b (ver sección 3.5) se han realizado agrupamientos con b igual a 0,5, 0,7 y 0,9.

La figura 6.20 muestra el BLEU y el WER obtenidos para este experimento.Como se puede observar, los resultados obtenidos utilizando clases y colocaciones no mejoran a los obtenidos sin clases. De nuevo, cuanto mayor es el número de clases más cercanos se encuentran tanto el BLEU como el WER de los obtenidos sin clases. Puede parecer que para los valores por encima de 3.000 clases se obtengan similares resultados a los obtenidos con OMEGA. Sin embargo, en este caso OMEGA se ha entrenado también con colocaciones y los resultados son ligeramente peores que cuando se entrena OMEGA sin colocaciones. En resumen, utilizando colocaciones tampoco mejoramos los resultados obtenidos con EuTrans II.

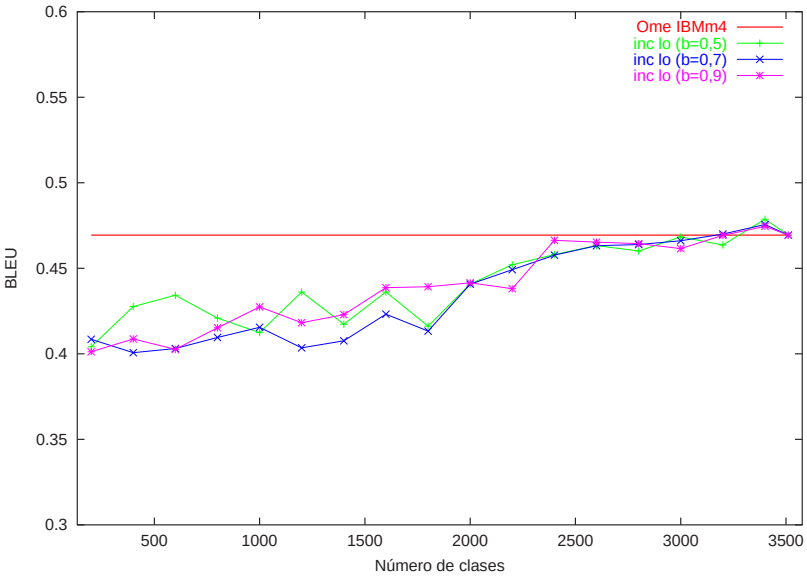


a)

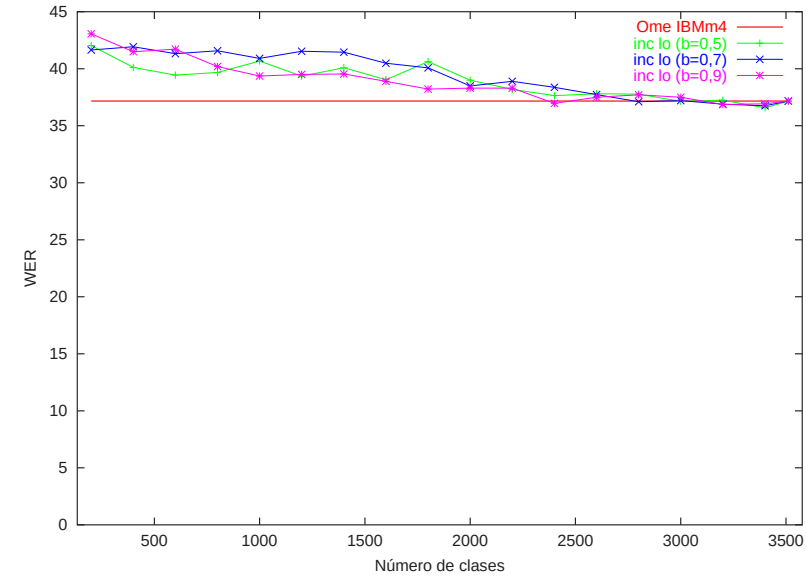


b)

Figura 6.19: a) BLEU y b) WER con OMEGA y OMEGA más las clases generadas por el algoritmo `incremental_lo` con distintos valores de b y distintos números de clases entre 200 y 2.869. (✍ pág. 199)



a)



b)

Figura 6.20: a) BLEU y b) WER con OMEGA y con OMEGA con las clases generadas por el algoritmo `incremental_lo` con distintos valores de b, colocaciones y distintos números de clases entre 200 y 3.511. (⇗ pág. 200)

Capítulo 7

Conclusiones

7.1. Conclusiones

Hemos propuesto en esta tesis un método de agrupamiento bilingüe, *e-cluster*, que tiene por objeto la mejora de sistemas de traducción automática basados en ejemplos (STBEs). En particular, hemos aplicado este agrupamiento para la mejora de STBEs basados en el modelo de transductores subsecuenciales (SSTs). Los SSTs han demostrado que pueden utilizarse satisfactoriamente en tareas de traducción de dominio restringido. Además, existen algoritmos eficientes, entre ellos OMEGA, que permiten inferir un SST a partir de un conjunto de ejemplos de traducción formados por pares de frases, una traducción de la otra (*corpus bilingüe*). El método propuesto, *e-cluster*, puede generar este agrupamiento bilingüe de forma automática utilizando la misma información que es necesaria para la inferencia del traductor. Es decir, no requiere de información adicional.

Este agrupamiento bilingüe no puede conseguirse simplemente mediante el agrupamiento de cada uno de los idiomas de forma independiente. Si se hiciera de esta forma, las clases obtenidas en ambos idiomas podrían no estar relacionadas. Y por tanto, el que supiéramos que una palabra del idioma origen pertenece a una determinada clase puede que no nos aportara ninguna información sobre en qué clase del idioma destino se encuentra su traducción. Un método de agrupamiento bilingüe debería agrupar las palabras de ambos idiomas de forma que existiera una correlación entre las clases en ambos idiomas. De esta forma, el agrupamiento podrá ser útil desde el punto de vista de la traducción.

Por otro lado, los algoritmos tradicionales de agrupamiento asignan cada palabra a una única clase. Sin embargo, es frecuente encontrar en el lenguaje palabras que presentan varios significados o que pueden realizar distintas fun-

ciones gramaticales. Por ejemplo, la palabra *banco* puede referirse a un asiento o a una entidad bancaria; y la palabra *ser* puede actuar como verbo y como sustantivo. Si el método de agrupamiento asigna cada palabra a una única clase, la agrupación resultante no podrá reflejar los distintos matices de cada palabra. Parece más adecuado realizar lo que se conoce como un *agrupamiento difuso*. Este agrupamiento asigna a cada palabra un grado de pertenencia a cada una de las clases posibles. De esta forma, se permite que una palabra pertenezca, en cierta medida, a más de una clase. Normalmente no se opta por este tipo de agrupamiento debido a que los algoritmos que lo implementan presentan un coste computacional más alto que los de agrupamiento convencional. De cualquier forma y desde un punto de vista de traducción sería interesante permitir que una palabra pudiera ser asignada a distintas clases, al menos, si se tradujera de forma distinta en el otro idioma dependiendo de su función gramatical o su significado.

El método *e-cluster* genera un agrupamiento bilingüe por medio del agrupamiento monolingüe de *palabras extendidas*. Hemos definido una palabra extendida como una palabra en el idioma destino que ha sido etiquetada con su traducción en el idioma origen. De esta forma, podemos diferenciar distintas acepciones o usos de una determinada palabra gracias a su traducción. Por ejemplo, las palabras extendidas $light_{encender}$ y $light_{luz}$ diferencian a la palabra inglesa *light* cuando ha sido traducción de *encender*, de cuando ha sido traducción de *luz*. Además, el uso de palabras extendidas nos permite contemplar en el lenguaje destino aquellas variaciones del lenguaje origen que no son propias del lenguaje destino. Por ejemplo, puede que el idioma origen distinga si un adjetivo es masculino o femenino y el idioma destino no; utilizando palabras extendidas podríamos diferenciar entre un adjetivo masculino o femenino en el idioma destino gracias a su traducción en el idioma origen. Así, el adjetivo en inglés *expensive* cuando es traducción del adjetivo *caro* en castellano, daría lugar a las palabras extendidas $expensive_{caro}$ o $expensive_{cara}$ dependiendo del género de cada caso.

A grandes rasgos, el método *e-cluster* consta de los siguientes pasos (ver sección 4.3). En primer lugar, se etiquetan las palabras del corpus destino con su traducción en el idioma origen. De esta forma, se obtiene un corpus de palabras extendidas. Sobre este nuevo corpus se realiza un agrupamiento monolingüe. El resultado obtenido es un agrupamiento de palabras extendidas. Cada clase de este agrupamiento estará formado, por tanto, por palabras en el idioma destino y la traducción con la que fueron etiquetadas. De esta forma, obtenemos pares de palabras en el idioma origen y destino agrupados en una misma clase bilingüe.

La idea tras estas clases de palabras extendidas es la siguiente. Sea c una clase de palabras extendidas. Sea una frase y su traducción, en las que aparecen la palabra origen x_i y la palabra destino y_i pertenecientes a c. Si sustituimos en la frase origen la palabra x_i por una palabra x_j que también forme parte de c, la traducción de la nueva frase vendría dada mediante la sustitución de la palabra y_i por la palabra y_j (donde $y_{j\ x_j}$ es una palabra extendida de c). Por ejemplo,

supongamos que tenemos la clase bilingüe $c = \{red_{rojo}, blue_{azul}\}$ y que sabemos que la frase «*el cuadrado rojo*» se traduce como «*the red square*». Entonces, el agrupamiento de palabras extendidas nos permite *aventurar* que la traducción de «*el cuadrado azul*», donde hemos sustituido *azul* por *rojo*, debería ser «*the blue square*» (que es el resultado de sustituir en la frase «*the red square*» la palabra *red* por *blue*).

El método propuesto presenta una de las cualidades deseables para los métodos de agrupamiento bilingüe: una gran correlación entre las clases en ambos idiomas. De hecho, la máxima posible, ya que asigna a una misma clase bilingüe palabras en ambos idiomas. De esta forma, sabiendo a qué clase bilingüe pertenece una determinada palabra origen, sabemos a qué clase pertenece su traducción —a la misma—.

Otra cualidad que también es deseable y presenta el método propuesto es que, pese a que cada palabra extendida se asigna a una única clase, una palabra del idioma origen o del idioma destino puede pertenecer a más de una clase si forma parte de más de una palabra extendida. Esto sucederá en aquellos casos en los que una palabra tiene varias traducciones. Por lo tanto, el método propuesto permite diferenciar aquellas palabras que son traducidas de diversas formas en función de su uso pudiendo asignarlas a clases distintas. No sólo esto, sino que además no incurre en un mayor coste computacional debido a que el método de agrupamiento monolingüe utilizado es un método de agrupamiento convencional (esto es, no es un método de agrupamiento difuso). La diferenciación se obtiene por medio de la creación de las palabras extendidas, no por el tipo de agrupamiento empleado.

Otra ventaja del método *e-cluster* consiste simplemente en que puede emplear cualquier algoritmo de agrupamiento monolingüe para generar un agrupamiento bilingüe. Tan sólo es necesario realizar ligeras modificaciones a los algoritmos de agrupamiento monolingüe para que puedan ser usados para el agrupamiento de palabras extendidas. De esto se desprende que las mejoras en el campo del agrupamiento monolingüe pueden trasladarse fácil y rápidamente para la mejora de agrupamientos bilingües.

El que las clases generadas sean clases de palabras extendidas presenta una ventaja adicional. Estas clases pueden integrarse fácilmente en el modelo de SSTs. Esta integración se lleva a cabo, a grandes rasgos, de la siguiente forma (ver sección 4.4). Una vez realizado el agrupamiento bilingüe, el corpus bilingüe se etiqueta con las clases obtenidas. Con este corpus etiquetado se infiere un SST. Paralelamente, se genera un autómata trivial para cada clase de palabras extendidas. Por último, el SST anteriormente generado, que contiene información de clases, puede expandirse utilizando los autómatas triviales de clases. Este SST expandido puede ser usado para la traducción de frases del idioma origen (ver sección 4.5). La traducción se obtiene mediante un sencillo post-proceso de la salida generada por el SST.

Puesto que *e-cluster* utiliza un algoritmo de agrupamiento monolingüe para la obtención de los agrupamientos bilingües hemos propuesto en esta tesis tres

algoritmos de agrupamiento monolingüe que son los que hemos utilizado para la implementación de *e-cluster* (ver capítulo 3). Los dos primeros, el *iterativo* y el *iterativo dejando uno fuera* (ver secciones 3.4 y 3.5), presentan un comportamiento similar al de otros métodos de agrupamiento monolingüe. Estos métodos parten de una distribución inicial de palabras en clases y mueven iterativamente cada palabra a cada posible clase buscando mejorar una función de optimización. Esta función es la información mutua entre clases adyacentes (ver sección 3.3). La diferencia entre ambos métodos es que el segundo utiliza la técnica de dejar uno fuera para calcular esta función simulando eventos no vistos y evitando de esta forma el sobreentrenamiento.

El tercero de los algoritmos de agrupamiento monolingüe propuestos, el *incremental*, puede verse como una extensión de los anteriores que intenta minimizar el efecto de la distribución inicial sobre el agrupamiento obtenido. Comienza con un número de clases menor que el deseado e incrementa el número de clases de una en una hasta alcanzarlo, decidiendo en cada incremento qué palabras se mueven en primer lugar a la nueva clase. Para cada nuevo número de clases realiza un agrupamiento iterativo pudiendo utilizar para este proceso cualquiera de los dos métodos anteriores. En la práctica, hemos utilizado este algoritmo únicamente en combinación con el *iterativo dejando uno fuera*, llamando a esta combinación el método *incremental dejando uno fuera*.

Por otro lado, los algoritmos de agrupamiento en general requieren que el número de clases en el que van a realizar el agrupamiento sea fijado de forma manual. Una característica que conviene destacar, por tanto, de los algoritmos de agrupamiento monolingüe que utilizan la técnica de *dejar uno fuera*, es que es posible utilizar su función de optimización para determinar de forma automática el número óptimo de clases. Los algoritmos *iterativo dejando uno fuera* e *incremental dejando uno fuera*, como se puede ver en los experimentos presentados en las secciones 6.2.1 y 6.3.1, pueden determinar de forma satisfactoria el número óptimo de clases en el que realizar el agrupamiento.

Volviendo al método *e-cluster* de agrupamiento bilingüe, este método presenta, en la forma en la que se plantea en el capítulo 4, la siguiente limitación: relaciona palabras en ambos idiomas una a una. Es una limitación en tanto en cuanto hay ocasiones en las que una determinada palabra en un idioma da lugar a más de una palabra en otro idioma. Llamamos *colocaciones* a estas secuencias de palabras que son traducción de otra palabra. Para mejorar *e-cluster* hemos propuesto en el capítulo 5 un algoritmo que detecta de forma automática estas colocaciones dado un corpus bilingüe. De esta forma, podemos generar un agrupamiento bilingüe utilizando *e-cluster* que tenga en cuenta posibles relaciones de varias palabras a una.

Además de mostrar un método para identificar colocaciones de forma automática, también hemos presentado en el capítulo 5 una técnica para evaluar varias posibles colocaciones de forma simultánea y, de esta forma, reducir el coste del proceso de evaluación. Puesto que sólo podremos evaluar de forma simultánea un conjunto de colocaciones que satisfagan una condición a la que

hemos denominado de *exclusión mutua* también hemos desarrollado un procedimiento que estima la idoneidad de cada posible colocación. Así, si varias colocaciones no satisfacen la condición de exclusión mutua podemos evaluar primero aquellas cuya estimación sea mejor. Esta estimación nos permitirá descartar directamente aquellas posibles colocaciones que no sean válidas. También hemos descrito cómo integrar estas colocaciones en el proceso de aprendizaje de un sistema de traducción basado en clases y el proceso seguido para llevar a cabo una traducción utilizando dicho sistema. En el capítulo 6 presentamos resultados experimentales utilizando colocaciones para la obtención de agrupamientos bilingües.

Los algoritmos de agrupamiento monolingüe y detección de colocaciones presentados en esta tesis se han utilizado para la mejora e implementación del método de agrupamiento bilingüe propuesto. Sin embargo, creemos que tienen aplicación y son interesantes por sí mismos en otras tareas de tratamiento de lenguaje natural.

Para comprobar las técnicas propuestas en esta tesis hemos realizado experimentos sobre dos corpora bilingües EuTrans I y EuTrans II (ver sección 6.1). Ambos corpora recogen situaciones de comunicación en un dominio limitado. Las situaciones que contemplan están entre las que pueden darse entre el cliente y el recepcionista de un hotel. El corpus EuTrans I es un corpus castellano–ingles relativamente sencillo del que se dispone de un adecuado número de frases de entrenamiento. Por contra, el corpus EuTrans II es un corpus italiano–inglés mucho más complejo, a la vez que más espontáneo, que el anterior. Posee un vocabulario más grande y un menor número de frases de entrenamiento. Además, posee numerosos errores tipográficos como palabras que aparecen a veces mal escritas o palabras que a veces aparecen juntas y otras veces separadas.

La primera serie de experimentos (ver sección 6.2) pretende estudiar el comportamiento de los algoritmos de agrupamiento monolingüe propuestos. Para ello, se han utilizado las partes en castellano y en inglés de los corpora EuTrans I y EuTrans II, respectivamente. Algunos de los experimentos realizados tienen por objeto comparar los métodos de agrupamiento propuestos entre sí y con algoritmos que representan el *estado del arte* en el agrupamiento monolingüe. Los restantes experimentos de agrupamiento monolingües pretenden comprobar si los algoritmos que utilizan la técnica *dejar uno fuera* pueden determinar de forma automática el número óptimo de clases.

Nuestras conclusiones a la vista de los resultados obtenidos sobre el corpus EuTrans I son las siguientes. El agrupamiento *incremental dejando uno fuera* obtiene los mejores agrupamientos monolingües de los métodos comparados y, además, presenta el comportamiento más estable con la variación del número de clases. En cuanto a los algoritmos *iterativo* e *iterativo dejando uno fuera*, éstos presentan un comportamiento similar al de los otros métodos de agrupamiento monolingüe comparados. Los métodos basados en la técnica de *dejar uno fuera* obtienen efectivamente una buena aproximación del número óptimo de clases.

En cuanto al corpus EuTrans II, puesto que ni los métodos de agrupamiento

presentados en esta tesis, ni los utilizados como referencia, obtienen resultados positivos, consideramos que el corpus es demasiado complejo para el número de ejemplos disponibles como para poder obtener un agrupamiento monolingüe adecuado razonable.

La segunda serie de experimentos (ver sección 6.3) pretende comprobar la influencia de los métodos propuestos de algoritmo bilingüe y colocaciones en la mejora de STBEs. Para ello se han realizado experimentos en los que se comparan los distintos modelos de alineamiento, los distintos métodos de agrupamiento bilingüe, el uso o no del agrupamiento bilingüe en la generación de los traductores, el uso o no de las colocaciones en la generación de los traductores y la utilización del mejor número de clases según el test frente al número óptimo de clases obtenido de forma automática. En el caso del corpus EuTrans I se han realizado experimentos para comprobar los anteriores supuestos con el corpus completo (10.000 frases) y con particiones de éste de entre 1.000 y 9.000 frases. De esta forma pretendemos estudiar el efecto de un menor número de muestras de entrenamiento en las técnicas propuestas. Los experimentos presentados con el corpus EuTrans II utilizan los métodos que se revelaron como mejores en los experimentos con EuTrans I.

Las conclusiones que hemos obtenido, a la vista de los experimentos realizados sobre el corpus EuTrans I, son las siguientes. La primera era predecible y se ve confirmada en los experimentos: cuanto mejor es el modelo de alineamiento, que se utiliza tanto para el agrupamiento bilingüe como para la inferencia del SST, mejor es el traductor inferido. Por otro lado, la comparación entre los métodos de agrupamiento bilingüe proporciona iguales resultados que los observados en el agrupamiento monolingüe: la versión bilingüe del método *incremental dejando uno fuera* genera mejores agrupamientos bilingües que el método *iterativo dejando uno fuera* y éste, a su vez, mejores que el método *iterativo*. Además, al igual que se vio en el caso monolingüe, el método *incremental dejando uno fuera* presenta un comportamiento más estable con la variación del número de clases. En cuanto al uso de colocaciones como paso previo al agrupamiento bilingüe, éste mejora sensiblemente la calidad de los traductores obtenidos. De hecho, usando el agrupamiento bilingüe sobre un corpus en el que se han identificado colocaciones para la inferencia de un SST, hemos obtenido mejores resultados que los reportados previamente utilizando categorías manuales. Por último, cuando hemos determinado de forma automática el número óptimo de clases, hemos obtenido resultados similares a los mejores obtenidos con distintos números de clases. Es decir, el número óptimo de clases determinado de forma automática por los algoritmos basados en la técnica de dejar uno fuera se puede utilizar para determinar el número de clases del agrupamiento.

En cuanto a los resultados experimentales, nos gustaría destacar que hemos disminuido la tasa de error de palabras (WER) obtenida por OMEGA de un 7 % a un 2,77 % utilizando OMEGA más agrupamiento bilingüe y colocaciones. Estando este último valor también por debajo del 3 % reportado previamente con categorías determinadas de forma manual.

Por contra, los resultados obtenidos con el corpus EuTrans II no han sido positivos. De nuevo, consideramos que estos resultados son debidos a la complejidad del corpus, que en este caso, y como hemos comentado antes en las conclusiones sobre el agrupamiento monolingüe, no permite ni siquiera la generación de un agrupamiento monolingüe razonable. De todas formas, nos gustaría emplear las técnicas propuestas en esta tesis en un corpus de similar complejidad al EuTrans II pero con un número adecuado de muestras de entrenamiento. Pese a que no han podido ser refrendadas en el corpus EuTrans II, consideramos que las conclusiones presentadas en esta sección son correctas.

7.2. Desarrollos futuros

Creemos que el trabajo presentado no acaba aquí, si no que más bien vemos esta tesis como el comienzo de una futura línea de investigación. Creemos, por tanto, que será posible encontrar mejoras de las técnicas propuestas, así como ampliar los métodos presentados. En esta sección describimos brevemente algunos de los desarrollos futuros que nos gustaría tener la oportunidad de llevar a cabo.

En primer lugar, nos gustaría estudiar la integración del método de agrupamiento bilingüe presentado en otros modelos de traducción automática; distintos a los SSTs. En particular, en modelos de traducción de estados finitos. Y de esta forma, poder evaluar la aportación de las técnicas presentadas a otros sistemas de traducción basados en ejemplos.

También nos parece interesante extender los métodos de agrupamiento monolingüe presentados para que realicen un *agrupamiento asimétrico*. Este método agrupa cada palabra de un corpus en dos agrupamientos diferenciados. Estos dos agrupamientos vienen marcados por la distinción, en el contexto de un modelo de lenguaje, entre la historia de una palabra y la palabra predicha por dicho modelo. Así, en el agrupamiento asimétrico, las palabras se agrupan en cada uno de los dos agrupamientos en función de su comportamiento como parte de una historia y como palabra predicha por el modelo de lenguaje, respectivamente. Nos parece interesante esta extensión debido a que este método parece ser que conduce a mejores agrupamientos monolingües [GGCL02] que los agrupamientos convencionales.

El algoritmo propuesto para la identificación de colocaciones realiza actualmente la identificación de éstas en sólo uno de los idiomas. Tal y como se describe en el capítulo 5 puede modificarse para la identificación de colocaciones en ambos idiomas. Esto plantea algunas dificultades en el proceso de traducción. Hemos pretendido mejorar la técnicas presentadas en esta tesis, entre ellas la detección de colocaciones, antes de abordar esta cuestión. Por ello, nos gustaría poder llevarla a cabo en futuros desarrollos.

Finalmente, otra mejora que nos gustaría realizar en el campo de identificación de colocaciones es la de permitir que palabras no consecutivas que se

comporten como una unidad desde el punto de vista de la traducción, sean tenidas en cuenta.

7.3. Publicaciones relacionadas con la tesis

Comentamos en primer lugar las publicaciones, que no siendo del autor, se consideran relevantes desde el punto de vista de la tesis por ser las que la motivaron. La primera de ellas es la tesis del Dr. Juan Miguel Vilar titulada «*Aprendizaje de traductores subsecuenciales para su empleo en tareas de dominio restringido*» [Vil98] en la que, entre otros aspectos del aprendizaje de traductores en dominios restringidos, se propone el algoritmo OMEGA. Otra fuente del mismo autor, en la que puede encontrarse la descripción del algoritmo OMEGA, es el artículo «*Improve the learning of subsequential transducers by using alignments and dictionaries*» [Vil00] publicado en *Lectures Notes in Artificial Intelligence*. En esta tesis hemos pretendido mejorar los traductores inferidos por este algoritmo mediante el agrupamiento bilingüe del corpus de entrenamiento.

En el artículo «*Spoken-language machine translation in limited domains: Can it be achieved by finite-state models?*» [VCJ+95] publicado en *Proceedings of the Theoretical and Methodological Issues in Machine Translation (TMI'95)* se apunta la posibilidad de que puedan mejorarse los SSTs mediante el uso de categorías. Posteriormente, en «*Using categories in the EuTrans system*» [ABC+97a] publicado en *Proceedings of the Spoken Language Translation Workshop, ACL and European Network in Language and Speech*, se describe el uso de categorías definidas de forma manual en el proyecto EuTrans y la mejora debido a éstas en los traductores generados.

Esta tesis presenta un método para la obtención de forma automática de agrupamientos bilingües que permitan la mejora de traductores automáticos. La primera publicación en la que se describe de forma breve la técnica propuesta inicialmente es «*Automatically deriving categories for translation*» [BV99a] publicada en *Proceedings of the Eurospeech'99*. Posteriormente, publicamos una versión ampliada de este artículo, «*Bilingual clustering using monolingual algorithms*» [BV99b] publicado en *Proceedings of the TMI'99*, en el que se describe, con más detalle, el proceso de agrupamiento bilingüe y su integración en los procesos de aprendizaje y traducción. Debido a la diferencia de orden entre las fechas para el envío de comunicaciones y las de celebración de ambos congresos, los artículos han sido finalmente publicados en el orden contrario al que fueron escritos.

En «*Improved bilingual clustering through monolingual clustering*» [BV99c] publicado en *Proceedings of the CAEPIA'99* presentamos los experimentos realizados con distintas distribuciones iniciales de palabras en clases para la obtención de mejores agrupamientos bilingües.

Finalmente, en «*Improving bilingual clustering by automatic compound discovery*» [BV01] publicado en *Proceedings of the IX Spanish Symposium on Pattern*

Recognition and Image Analysis presentamos una primera aproximación a lo que hemos denominado en esta tesis colocaciones.

Además, hemos colaborado en la siguiente publicación, relacionada con la traducción pero no con el agrupamiento bilingüe, «*Some approaches to statistical and finite-state speech-to-speech translation*» [CNO+] que está pendiente de aceptación para su publicación en la revista *Computer Speech and Language*. Este artículo presenta una comparativa de varias aproximaciones a la traducción automática mediante métodos estadísticos y de estados finitos.

Bibliografía

[ABC+97a] Amengual, J. C., J. M. Benedí, F. Casacuberta, A. Castaño, A. Castellanos, D. Llorens, A. Marzal, F. Prat, E. Vidal y J. M. Vilar (1997). *Using categories in the EuTrans system.* En *Proceedings of the Spoken Language Translation Workshop, ACL and European Network in Language and Speech*, págs. 44–53. Madrid (Spain).

[ABC+97b] Amengual, Juan C., José M. Benedí, Francisco Casacuberta, Asunción Castaño, Antonio Castellanos, David Llorens, Andrés Marzal, Federico Prat, Enrique Vidal y Juan M. Vilar (1997). *Error correcting parsing for text-to-text machine translation using finite state models.* En *Proceedings of Theoretical and Methodological Issues in Machine Translation (TMI'97)*, págs. 135–142. Santa Fe, New Mexico (USA).

[ACDC99] Aiello, D., L. Cerrato, C. Delogu y A. di Carlo (1999). *The acquisition of a speech corpus for limited domain translation.* En *European Conference on Speech Communication and Technology*, tomo 5, págs. 2223–2226. Budapest (Hungary).

[BDd+92] Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai y Robert L. Mercer (1992). *Class-based n-gram models of natural language.* Computational Linguistics, tomo 18(4): págs. 467–479.

[BDDM93] Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra y Robert L. Mercer (junio 1993). *The mathematics of machine translation: Parameter estimation.* Computational Linguistics, tomo 19(2): págs. 263–312.

[Ber79] Berstel, J. (1979). *Transductions and Context-Free Languages.* Teubner Verlag.

[Bru78] Brucker, P. (1978). *On the complexity of clustering problems.* En R. Henn, B. Korte y W. Oettli (eds.), *Optimization and Operations Research*, tomo 157 de *Lecture Notes in Economics and Mathematica Systems*, págs. 45–54.

[BV99a] Barrachina, Sergio y Juan Miguel Vilar (septiembre 1999). *Automatically deriving categories for translation.* En *Proceedings of the European Conference on Speech Communication and Technology (EuroSpeech'99)*, págs. 2415–2418. Budapest (Hungary). ISSN 1018-4074.

[BV99b] Barrachina, Sergio y Juan Miguel Vilar (agosto 1999). *Bilingual clustering using monolingual algorithms.* En *Proceedings of Theoretical and Methodological Issues in Machine Translation (TMI'99)*, págs. 77–87. Chester (U.K.).

[BV99c] Barrachina, Sergio y Juan Miguel Vilar (noviembre 1999). *Improved bilingual clustering through monolingual clustering.* En *Proceedings of the Spanish Association for Artificial Intelligence (CAEPIA'99)*, tomo 1, págs. 183–190. Murcia (Spain). ISBN 931170-0-5.

[BV01] Barrachina, Sergio y Juan Miguel Vilar (mayo 2001). *Improving bilingual clustering by automatic compound discovery.* En *Proceedings of the IX Spanish Symposium on Pattern Recognition and Image Analysis*, tomo 2, págs. 373–378. Benicasim (Spain). ISBN 84-8021-351-5.

[CC97] Castaño, M. A. y F. Casacuberta (septiembre 1997). *A connectionist approach to machine translation.* En G. Kokkinakis, N. Fakotakis y E. Dermatas (eds.), *Proceedings of the European Conference on Speech Communication and Technology (EuroSpeech'97)*, tomo 1, págs. 91–94. ESCA, Rodas (Greece).

[CG98] Chen, Stanley F. y Joshua Goodman (1998). *An empirical study of smoothing techniques for language modeling.* Informe técnico, Harvard University.

[CGV94] Castellanos, A., I. Galiano y E. Vidal (septiembre 1994). *Application of OSTIA to machine translation tasks.* En Rafael C. Carrasco y José Oncina (eds.), *Grammatical Inference and Applications*, tomo 862 de *Lecture Notes in Computer Science*, págs. 93–105. Springer-Verlag.

[CNO$^+$] Casacuberta, F., H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C. Martínez, S. Molau, F. Nevado, M. Pastor, D. Picó, A. Sanchís y C. Tillman. *Some approaches*

to statistical and finite-state speech-to-speech translation. Computer Speech and Language. Enviado, pendiente de aceptación.

[FW90] Fung, Pascale y Dekai Wu (1990). *Coerced markov models for cross-lingual lexical-tag relations.* En *Proceedings of Theoretical and Methodological Issues in Machine Translation (TMI'95)*, págs. 240–255. Leuven (Belgium).

[GGCL02] Gao, Jianfeng, Joshua T. Goodman, Guichong Cao y Hang Li (julio 2002). *Exploring asymmetric clustering for statistical language modeling.* En *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'2002)*, págs. 183–190. Philadelphia (USA).

[Hut99] Hutchins, John (agosto 1999). *Introduction to Machine Translation Tools and Computerized Translation Tools.* EAMT Tutorial at TMI'99.

[IFRZ00] Instituto Tecnológico de Informática, Fondazione Ugo Bordoni, Rheinisch-Westfälische Technische Hochschule Aachen Lehrstuhl für Informatik VI y ZERES GmbH Bochum (septiembre 2000). *Example-Based Language Translation Systems (EuTrans). Final Report.* Informe técnico, Information Technology Long Term Research Domain Open Scheme. Project number 30268.

[JC99] Juan Ciscar, Alfons (1999). *Optimización de prestaciones en técnicas de aprendizaje no supervisado y su aplicación al reconocimiento de formas.* Tesis Doctoral, Universidad Politécnica de Valencia, Valencia (Spain).

[KN93] Kneser, Reinhard y Hermann Ney (1993). *Improved clustering techniques for class-based statistical language modelling.* En *Proceedings of the European Conference on Speech Communication and Technology (EuroSpeech'93)*, tomo 2, págs. 973–976. Berlin (Germany).

[Kni97] Knight, Kevin (1997). *Automating knowledge acquisition for machine translation.* AI Magazine, tomo 18(4): págs. 81–96.

[Lee97] Lee, Lillian Jane (mayo 1997). *Similarity-Based Approaches to Natural Language Processing.* Tesis Doctoral, Harvard University, Cambridge, Massachusetts (USA).

[Mel97] Melamed, I. Dan (1997). *Automatic discovery of non-compositional compounds in parallel data.* En *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP'97)*, págs. 97–108. Association for Computational Linguistics, Somerset, New Jersey (USA)).

[MLN95] Martin, Sven, Jörg Liermann y Hermann Ney (1995). *Algorithms for bigram and trigram word clustering.* En *Proceedings of the European Conference on Speech Communication and Technology (EuroSpeech'95)*, págs. 1253–1256. Madrid (Spain).

[MS00] Manning, Christopher D. y Hinrich Schütze (2000). *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts (USA), 1 edición.

[NE93] Ney, Hermann y Ute Essen (1993). *Estimating 'small' probabilities by leaving-one-out.* En *Proceedings of the European Conference on Speech Communication and Technology (EuroSpeech'93)*, tomo 3, págs. 2239–2242.

[Nie97] Niesler, Thomas (junio 1997). *Category-based statistical language models.* Tesis Doctoral, University of Cambridge, Cambridge (U.K.).

[Och99] Och, Franz Josef (junio 1999). *An efficient method for determining bilingual word classes.* En *Proceedings of the 9th Conferecence of the European Chapter of the Association for Computational Linguistics (EACL'99)*, págs. 71–76. Bergen (Norway).

[OGV93] Oncina, J., P. García y E. Vidal (1993). *Learning subsequential transducers for pattern recognition interpretation tasks.* IEEE Transactions on Pattern Analysis and Machine Intelligence, tomo 15(5): págs. 448–458.

[ON00] Och, F. J. y H. Ney (octubre 2000). *Improved statistical alignment models.* En *38th Annual Meeting of the Association for Computational Linguistics (ACL'2000)*, págs. 440–447. Hong Kong (China).

[OTN99] Och, Franz Josef, Christoph Tillmann y Hermann Ney (junio 1999). *Improved alignment models for statistical machine translation.* En *Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora*, págs. 20–28.

[OW98] Och, Franz Josef y Hans Weber (1998). *Improving statistical natural language translation with categories and rules.* En *Proceedings of the International Conference on Computational Linguistics (COLING'98)*, págs. 985–989. Montreal (Canada).

[PRWZ01] Papineni, Kishore, Salim Roukos, Todd Ward y Wei-Jing Zhu (17 septiembre 2001). *BLEU: a method for automatic evaluation of machine translation.* Informe técnico, IBM Research Division.

[TK99] Theodoridis, Sergios y Konstantinos Koutroumbas (1999). *Pattern Recognition.* Academic Press.

[VCJ+95] Vilar, J. M., A. Castellanos, V. Jimenez, J. Oncina, H. Rulot, J. A. Sánchez y E. Vidal (1995). *Spoken-language machine translation in limited domains: Can it be achieved by finite-state models?* En *Proceedings of Theoretical and Methodological Issues in Machine Translation (TMI'95)*, págs. 326–333. Leuven (Belgium).

[Vil98] Vilar, Juan Miguel (1998). *Aprendizaje de traductores subsecuenciales para su empleo en tareas de dominio restringido.* Tesis Doctoral, Dpto. de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia, Valencia (Spain).

[Vil00] Vilar, J. M. (2000). *Improve the learning of subsequential transducers by using alignments and dictionaries.* En *Grammatical Inference: Algorithms and Applications*, tomo 1891 de *Lecture Notes in Artificial Intelligence*, págs. 298–312. Springer-Verlag.

[VJA+96] Vilar, J. M., V. M. Jiménez, J. C. Amengual, A. Castellanos, D. Llorens y E. Vidal (1996). *Text and speech translations by means of subsequential transducers.* Natural Language Engineering, tomo 2(4): págs. 351–354.

[WLW96] Wang, Ye-Yi, John Lafferty y Alex Waibel (1996). *Word clustering with parallel spoken language corpora.* En *Proceedings of the International Conference on Spoken Language Processing (ICSLP'96)*, tomo 4, págs. 2364–2367. Philadelphia (USA).

[YB97] Young, Steve y Gerrit Bloothooft (eds.) (1997). *Corpus-based Methods in Language and Speech Processing.* Kluwer Academic.

APÉNDICE A

Ejemplos de agrupamientos y colocaciones

A.1. Introducción

Presentamos en este apéndice dos ejemplos completos de agrupamiento bilingüe utilizando la técnica de palabras extendidas propuesta en el capítulo 4. El primero de ellos corresponde al agrupamiento bilingüe del corpus EuTrans I con el número de clases óptimo determinado de forma automática. El segundo corresponde al agrupamiento bilingüe del mismo corpus pero después de la detección automática de colocaciones. El número de clases también es el determinado de forma automática. Además, mostramos el conjunto de colocaciones detectadas por medio del algoritmo propuesto en el capítulo 5.

Las clases mostradas son clases de palabras extendidas. En los ejemplos, estas palabras extendidas son palabras inglesas etiquetadas con su traducción al castellano. Cada clase agrupa, por tanto, a una serie de palabras en el idioma origen (castellano) y a una serie de palabras en el idioma destino (inglés).

A.2. Agrupamiento de EuTrans I

El siguiente cuadro muestra el agrupamiento obtenido sobre el corpus EuTrans I utilizando el algoritmo incremental dejando uno fuera (ver sección 3.6). El número de clases utilizado para el agrupamiento corresponde al número de clases óptimo determinado de forma automática. Como método de alineamiento

se ha utilizado el modelo IBMm4. Se puede ver una serie de comentarios sobre algunas de las clases que forman este agrupamiento en la sección 4.3.3.

En la columna de la izquierda se muestra el número con el que se ha etiquetado cada una de las clases obtenidas. En la columna de la derecha se muestran las palabras extendidas que forman cada clase.

1 :	would$_{le}$ would$_{les}$ can$_{podría}$ could$_{podrían}$ can$_{puede}$ can$_{pueden}$ will$_{quiere}$ will$_{quieren}$ do$_{tiene}$
2 :	?$_{?}$
3 :	want$_{deseo}$ want$_{quiero}$
4 :	send$_{llevar}$ carry$_{llevarnos}$ send$_{llevarnos}$ put$_{subir}$ put$_{subirnos}$
5 :	give$_{dé}$ give$_{den}$ wake$_{despertara}$ wake$_{despertaran}$ wake$_{despertase}$ wake$_{despertasen}$ wake$_{despierte}$ wake$_{despierten}$ give$_{diera}$ give$_{dieran}$ give$_{diese}$ give$_{diesen}$
6 :	fourteenth$_{catorce}$ fifth$_{cinco}$ fourth$_{cuatro}$ nineteenth$_{diecinueve}$ eighteenth$_{dieciocho}$ sixteenth$_{dieciséis}$ seventeenth$_{diecisiete}$ tenth$_{diez}$ twelfth$_{doce}$ second$_{dos}$ ninth$_{nueve}$ eighth$_{ocho}$ eleventh$_{once}$ fifteenth$_{quince}$ sixth$_{seis}$ seventh$_{siete}$ thirteenth$_{trece}$ thirtieth$_{treinta}$ thirty-first$_{treinta}$ third$_{tres}$ first$_{uno}$ twentieth$_{veinte}$ twenty-fifth$_{veinticinco}$ twenty-fourth$_{veinticuatro}$ twenty-second$_{veintidós}$ twenty-ninth$_{veintinueve}$ twenty-eighth$_{veintiocho}$ twenty-sixth$_{veintiséis}$ twenty-seventh$_{veintisiete}$ twenty-third$_{veintitrés}$ twenty-first$_{veintiuno}$
7 :	Ángel$_{Ángel}$ Alicia$_{Alicia}$ Amelia$_{Amelia}$ Ana$_{Ana}$ Andrés$_{Andrés}$ Beatriz$_{Beatriz}$ César$_{César}$ Carmen$_{Carmen}$ Celestino$_{Celestino}$ Concepción$_{Concepción}$ Cristina$_{Cristina}$ Daniel$_{Daniel}$ Eduardo$_{Eduardo}$ Emilio$_{Emilio}$ Eva$_{Eva}$ Federico$_{Federico}$ Gloria$_{Gloria}$ Gregorio$_{Gregorio}$ Ignacio$_{Ignacio}$ Inmaculada$_{Inmaculada}$ Isabel$_{Isabel}$ Jacinto$_{Jacinto}$ Javier$_{Javier}$ Jesús$_{Jesús}$ Joaquín$_{Joaquín}$ Jorge$_{Jorge}$ Juan$_{Juan}$ Julia$_{Julia}$ Julio$_{Julio}$ Lidia$_{Lidia}$ Luis$_{Luis}$ Manuel$_{Manuel}$ Manuela$_{Manuela}$ Margarita$_{Margarita}$ Marina$_{Marina}$ Micaela$_{Micaela}$ Pilar$_{Pilar}$ Rafael$_{Rafael}$ Rosalía$_{Rosalía}$ Rosario$_{Rosario}$ Santiago$_{Santiago}$ Sergio$_{Sergio}$ Sonia$_{Sonia}$ Susana$_{Susana}$ Teresa$_{Teresa}$ Tomás$_{Tomás}$ Victoria$_{Victoria}$ Virginia$_{Virginia}$
8 :	you$_{¿}$
9 :	leaving$_{iremos}$ leaving$_{marchamos}$ leaving$_{vamos}$
10 :	with$_{con}$ with$_{tenga}$

continúa en la página siguiente ➪

➪ viene de la página anterior

11 :	my $_{\text{los}}$ our $_{\text{los}}$ our $_{\text{nuestros}}$
12 :	Álvarez $_{\text{Álvarez}}$ Aguilera $_{\text{Aguilera}}$ Alted $_{\text{Alted}}$ Aragón $_{\text{Aragón}}$ Arenas $_{\text{Arenas}}$ Arnau $_{\text{Arnau}}$ Arroyo $_{\text{Arroyo}}$ Balaguer $_{\text{Balaguer}}$ Ballester $_{\text{Ballester}}$ Barberá $_{\text{Barberá}}$ Barber $_{\text{Barber}}$ Belenguer $_{\text{Belenguer}}$ Bellver $_{\text{Bellver}}$ Beltrán $_{\text{Beltrán}}$ Berrueco $_{\text{Berrueco}}$ Betoret $_{\text{Betoret}}$ Bordons $_{\text{Bordons}}$ Borillo $_{\text{Borillo}}$ Borrás $_{\text{Borrás}}$ Botella $_{\text{Botella}}$ Cabedo $_{\text{Cabedo}}$ Cabo $_{\text{Cabo}}$ Cabrera $_{\text{Cabrera}}$ Calatayud $_{\text{Calatayud}}$ Calleja $_{\text{Calleja}}$ Campo $_{\text{Campo}}$ Cantero $_{\text{Cantero}}$ Carpio $_{\text{Carpio}}$ Cerezo $_{\text{Cerezo}}$ Colomer $_{\text{Colomer}}$ Cornelles $_{\text{Cornelles}}$ Cutillas $_{\text{Cutillas}}$ Díaz $_{\text{Díaz}}$ Edo $_{\text{Edo}}$ Escrig $_{\text{Escrig}}$ Espinosa $_{\text{Espinosa}}$ Fernández $_{\text{Fernández}}$ Ferrando $_{\text{Ferrando}}$ Fuster $_{\text{Fuster}}$ Gómez $_{\text{Gómez}}$ Gallego $_{\text{Gallego}}$ García $_{\text{García}}$ Gil $_{\text{Gil}}$ Gimeno $_{\text{Gimeno}}$ Gual $_{\text{Gual}}$ Guardiola $_{\text{Guardiola}}$ Guijarro $_{\text{Guijarro}}$ Gumbau $_{\text{Gumbau}}$ Herrero $_{\text{Herrero}}$ Ibáñez $_{\text{Ibáñez}}$ Iborra $_{\text{Iborra}}$ Ivars $_{\text{Ivars}}$ Izquierdo $_{\text{Izquierdo}}$ Jiménez $_{\text{Jiménez}}$ López $_{\text{López}}$ Llorens $_{\text{Llorens}}$ Lobo $_{\text{Lobo}}$ Maestro $_{\text{Maestro}}$ Marqués $_{\text{Marqués}}$ Martí $_{\text{Martí}}$ Martínez $_{\text{Martínez}}$ Meda $_{\text{Meda}}$ Mestre $_{\text{Mestre}}$ Mingot $_{\text{Mingot}}$ Mira $_{\text{Mira}}$ Miralles $_{\text{Miralles}}$ Moliner $_{\text{Moliner}}$ Monferrer $_{\text{Monferrer}}$ Montero $_{\text{Montero}}$ Montoro $_{\text{Montoro}}$ Morales $_{\text{Morales}}$ Moreno $_{\text{Moreno}}$ Moya $_{\text{Moya}}$ Nebot $_{\text{Nebot}}$ Orenga $_{\text{Orenga}}$ Ortiz $_{\text{Ortiz}}$ Pérez $_{\text{Pérez}}$ Paches $_{\text{Paches}}$ Padilla $_{\text{Padilla}}$ Pallarés $_{\text{Pallarés}}$ Pastor $_{\text{Pastor}}$ Peinado $_{\text{Peinado}}$ Peris $_{\text{Peris}}$ Peñarroja $_{\text{Peñarroja}}$ Piquer $_{\text{Piquer}}$ Pitarch $_{\text{Pitarch}}$ Quereda $_{\text{Quereda}}$ Querol $_{\text{Querol}}$ Ríos $_{\text{Ríos}}$ Ródenas $_{\text{Ródenas}}$ Ramírez $_{\text{Ramírez}}$ Ramos $_{\text{Ramos}}$ Redondo $_{\text{Redondo}}$ Revilla $_{\text{Revilla}}$ Rivera $_{\text{Rivera}}$ Roda $_{\text{Roda}}$ Rodríguez $_{\text{Rodríguez}}$ Romero $_{\text{Romero}}$ Royo $_{\text{Royo}}$ Rubio $_{\text{Rubio}}$ Sáez $_{\text{Sáez}}$ Salsas $_{\text{Salsas}}$ Sanfeliú $_{\text{Sanfeliú}}$ Santos $_{\text{Santos}}$ Segura $_{\text{Segura}}$ Serrano $_{\text{Serrano}}$ Sevilla $_{\text{Sevilla}}$ Silvestre $_{\text{Silvestre}}$ Suárez $_{\text{Suárez}}$ Tena $_{\text{Tena}}$ Vázquez $_{\text{Vázquez}}$ Valls $_{\text{Valls}}$ Varela $_{\text{Varela}}$ Velasco $_{\text{Velasco}}$ Viciano $_{\text{Viciano}}$ Vidal $_{\text{Vidal}}$ Vilanova $_{\text{Vilanova}}$ Villanueva $_{\text{Villanueva}}$
13 :	another $_{\text{otra}}$
14 :	there $_{\text{existe}}$
15 :	is $_{\text{es}}$
16 :	bath $_{\text{aseo}}$ minibar $_{\text{bar}}$ bath $_{\text{baño}}$ bathroom $_{\text{baño}}$ shower $_{\text{ducha}}$ safe $_{\text{fuerte}}$ telephone $_{\text{teléfono}}$ tv $_{\text{televisión}}$
17 :	after $_{\text{cargo}}$ carrying $_{\text{llevar}}$ sending $_{\text{llevar}}$ carrying $_{\text{llevarme}}$ sending $_{\text{llevarme}}$ carrying $_{\text{llevarnos}}$ sending $_{\text{llevarnos}}$ putting $_{\text{subir}}$ putting $_{\text{subirme}}$ putting $_{\text{subirnos}}$
18 :	fourteen $_{\text{catorce}}$ nineteen $_{\text{diecinueve}}$ eighteen $_{\text{dieciocho}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

	sixteen dieciséis seventeen diecisiete fifteen quince thirteen trece thirty treinta twenty veinte twenty-five veinticinco twenty-four veinticuatro twenty-two veintidós twenty-nine veintinueve twenty-eight veintiocho twenty-seven veintisiete twenty-three veintitrés twenty-one veintiuno
19 :	in en
20 :	okay acuerdo okay conforme okay correcto sorry disculpe hello hola no no sorry perdón sorry perdone yes sí
21 :	mistake se
22 :	. .
23 :	room habitación
24 :	five cinco four cuatro nine nueve eight ocho six seis seven siete three tres
25 :	booked reservado
26 :	four catorce nine diecinueve six dieciséis seven diecisiete oh diez two doce five quince three trece
27 :	morning mañana evening noche afternoon tarde
28 :	at las
29 :	included anotados included apuntados included incluidos
30 :	has han
31 :	today hoy
32 :	would desearíamos would querríamos would quisiéramos
33 :	double doble single individual
34 :	my mi
35 :	key llave
36 :	we hemos we tenemos
37 :	Sunday domingo Thursday jueves Monday lunes Tuesday martes Wednesday miércoles Saturday sábado Friday viernes
38 :	want deseamos want queremos
39 :	quiet tranquila
40 :	double dobles single individuales
41 :	does cuesta
42 :	send llevara send llevaran send llevase send llevasen carry lleve send lleve send lleven put suba

continúa en la página siguiente ➪

➪ viene de la página anterior

43 :	the $_{\text{la}}$
44 :	to $_{\text{a}}$
45 :	give $_{\text{dar}}$ wake $_{\text{despertar}}$
46 :	reservation $_{\text{reserva}}$
47 :	me $_{\text{me}}$ us $_{\text{nos}}$
48 :	two $_{\text{dos}}$
49 :	to $_{\text{de}}$
50 :	included $_{\text{anotado}}$ included $_{\text{apuntado}}$
51 :	on $_{\text{el}}$
52 :	half $_{\text{media}}$
53 :	a $_{\text{una}}$
54 :	taxi $_{\text{taxi}}$
55 :	days $_{\text{días}}$ nights $_{\text{noches}}$ weeks $_{\text{semanas}}$
56 :	ask $_{\text{pedir}}$ ask $_{\text{pedirme}}$ ask $_{\text{pedirnos}}$
57 :	after $_{\text{pasado}}$
58 :	number $_{\text{número}}$
59 :	would $_{\text{desearía}}$ would $_{\text{quisiera}}$
60 :	rooms $_{\text{habitaciones}}$
61 :	and $_{\text{y}}$
62 :	for $_{\text{de}}$
63 :	there $_{\text{hay}}$
64 :	luggage $_{\text{equipaje}}$
65 :	travel $_{\text{viaje}}$
66 :	like $_{\text{gustaría}}$
67 :	have $_{\text{he}}$
68 :	town $_{\text{ciudad}}$ mountain $_{\text{montaña}}$
69 :	book $_{\text{reservar}}$
70 :	in $_{\text{de}}$
71 :	breakfast $_{\text{desayuno}}$
72 :	quarter $_{\text{cuarto}}$
73 :	that $_{\text{parece}}$
74 :	available $_{\text{libre}}$
75 :	for $_{\text{para}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

76 :	air $_{\text{aire}}$
77 :	conditioning $_{\text{acondicionado}}$
78 :	send $_{\text{baja}}$ call $_{\text{llama}}$
79 :	bill $_{\text{cuenta}}$ bill $_{\text{factura}}$
80 :	the $_{\text{las}}$
81 :	bags $_{\text{bolsas}}$
82 :	ten $_{\text{diez}}$ twelve $_{\text{doce}}$ eleven $_{\text{once}}$
83 :	? $_{\text{¿}}$
84 :	pay $_{\text{abonar}}$ pay $_{\text{pagar}}$
85 :	should $_{\text{he}}$ should $_{\text{tengo}}$
86 :	please $_{\text{por}}$
87 :	tomorrow $_{\text{mañana}}$
88 :	are $_{\text{vamos}}$
89 :	Miss $_{\text{señorita}}$
90 :	five $_{\text{cincuenta}}$ four $_{\text{cuarenta}}$ nine $_{\text{noventa}}$ eight $_{\text{ochenta}}$ six $_{\text{sesenta}}$ seven $_{\text{setenta}}$ three $_{\text{treinta}}$
91 :	how $_{\text{cuánta}}$ how $_{\text{cuánto}}$
92 :	look $_{\text{hacerse}}$ mind $_{\text{importaría}}$
93 :	our $_{\text{nuestro}}$
94 :	luggage $_{\text{bultos}}$
95 :	! $_{\text{¡}}$
96 :	give $_{\text{darnos}}$ wake $_{\text{despertarnos}}$
97 :	would $_{\text{querría}}$
98 :	two $_{\text{veintidós}}$
99 :	check $_{\text{cheques}}$
100 :	give $_{\text{darme}}$ wake $_{\text{despertarme}}$
101 :	been $_{\text{equivocado}}$
102 :	keys $_{\text{llaves}}$
103 :	until $_{\text{hasta}}$
104 :	view $_{\text{vistas}}$
105 :	good $_{\text{buena}}$
106 :	mistake $_{\text{equivocación}}$
107 :	one $_{\text{una}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

108 :	am $_{\text{voy}}$
109 :	extras $_{\text{extras}}$ expenses $_{\text{gastos}}$ taxes $_{\text{impuestos}}$
110 :	error $_{\text{error}}$
111 :	registration $_{\text{hoja}}$
112 :	give $_{\text{déme}}$ give $_{\text{denme}}$ wake $_{\text{despiérteme}}$ wake $_{\text{despiértenme}}$
113 :	problem $_{\text{problema}}$
114 :	you $_{\text{nos}}$
115 :	all $_{\text{todos}}$
116 :	hot $_{\text{calurosa}}$ expensive $_{\text{caras}}$ cold $_{\text{fría}}$ noisy $_{\text{ruidosa}}$
117 :	week $_{\text{semana}}$
118 :	call $_{\text{llamar}}$ call $_{\text{llamarnos}}$
119 :	full $_{\text{completa}}$
120 :	this $_{\text{esta}}$
121 :	explain $_{\text{detalle}}$ explain $_{\text{explique}}$ prepare $_{\text{prepare}}$ check $_{\text{repase}}$
122 :	bus $_{\text{autobús}}$ car $_{\text{coche}}$ date $_{\text{es}}$ date $_{\text{estamos}}$
123 :	giving $_{\text{darme}}$ waking $_{\text{despertarme}}$ showing $_{\text{mostrarme}}$
124 :	very $_{\text{muy}}$
125 :	made $_{\text{hice}}$
126 :	could $_{\text{podría}}$
127 :	agree $_{\text{estoy}}$ matter $_{\text{importa}}$
128 :	suitcases $_{\text{maletas}}$
129 :	not $_{\text{no}}$
130 :	hot $_{\text{calor}}$ cold $_{\text{frío}}$
131 :	bag $_{\text{bolsa}}$
132 :	one $_{\text{ciento}}$ four $_{\text{cuatrocientos}}$ two $_{\text{doscientos}}$ nine $_{\text{novecientos}}$ eight $_{\text{ochocientos}}$ five $_{\text{quinientos}}$ six $_{\text{seiscientos}}$ seven $_{\text{setecientos}}$ three $_{\text{trescientos}}$
133 :	o'clock $_{\text{punto}}$
134 :	reception $_{\text{recepción}}$
135 :	view $_{\text{vista}}$
136 :	a $_{\text{alguna}}$
137 :	was $_{\text{produjo}}$
138 :	fill $_{\text{rellenar}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

139 :	our $_{\text{nuestra}}$
140 :	call $_{\text{llame}}$
141 :	one $_{\text{uno}}$
142 :	name $_{\text{nombre}}$
143 :	made $_{\text{hecha}}$
144 :	move $_{\text{cambiarme}}$
145 :	give $_{\text{dénos}}$ give $_{\text{dennos}}$ wake $_{\text{despiértennos}}$ wake $_{\text{despiértenos}}$
146 :	next $_{\text{próximo}}$
147 :	my $_{\text{mis}}$
148 :	think $_{\text{creo}}$ think $_{\text{pienso}}$
149 :	changing $_{\text{cambiarnos}}$
150 :	make $_{\text{hágame}}$ make $_{\text{háganos}}$
151 :	have $_{\text{tengo}}$
152 :	tonight $_{\text{esta}}$
153 :	station $_{\text{estación}}$
154 :	change $_{\text{cambiar}}$ change $_{\text{nuestra}}$
155 :	booked $_{\text{reservadas}}$
156 :	day $_{\text{día}}$
157 :	thank $_{\text{gracias}}$ could $_{\text{llámeme}}$ could $_{\text{llámenos}}$
158 :	made $_{\text{hecho}}$
159 :	name $_{\text{llamo}}$
160 :	warmer $_{\text{cálida}}$ warmer $_{\text{fría}}$ quieter $_{\text{más}}$ quieter $_{\text{ruido}}$ quieter $_{\text{ruidosa}}$
161 :	Mrs $_{\text{señora}}$
162 :	what $_{\text{qué}}$
163 :	booked $_{\text{reservamos}}$
164 :	leave $_{\text{irme}}$ leave $_{\text{marcharme}}$
165 :	send $_{\text{súbanme}}$ send $_{\text{súbannos}}$ send $_{\text{súbanos}}$ send $_{\text{suba}}$ send $_{\text{suban}}$
166 :	water $_{\text{caliente}}$
167 :	our $_{\text{nuestras}}$
168 :	been $_{\text{producido}}$
169 :	look $_{\text{vistazo}}$
170 :	card $_{\text{crédito}}$ cards $_{\text{crédito}}$
171 :	has $_{\text{ha}}$

continúa en la página siguiente ➪

✑ viene de la página anterior

172 :	explain $_{\text{detalla}}$ explain $_{\text{explica}}$ prepare $_{\text{prepara}}$ check $_{\text{repasa}}$
173 :	do $_{\text{tienen}}$
174 :	everything $_{\text{todo}}$
175 :	tax $_{\text{valor}}$
176 :	booked $_{\text{reservada}}$
177 :	including $_{\text{incluyendo}}$
178 :	board $_{\text{pensión}}$
179 :	send $_{\text{subir}}$ send $_{\text{subirme}}$ send $_{\text{subirnos}}$
180 :	taking $_{\text{echara}}$
181 :	down $_{\text{bájenme}}$ down $_{\text{bájennos}}$ down $_{\text{bájenos}}$ down $_{\text{bajar}}$ down $_{\text{bajara}}$ down $_{\text{bajarme}}$ down $_{\text{bajarnos}}$ down $_{\text{bajase}}$ down $_{\text{bajasen}}$ down $_{\text{baje}}$ down $_{\text{bajen}}$ up $_{\text{subiera}}$
182 :	giving $_{\text{darnos}}$ waking $_{\text{despertarnos}}$ showing $_{\text{mostrarnos}}$
183 :	very $_{\text{mucho}}$
184 :	suitcase $_{\text{maleta}}$
185 :	, $_{,}$
186 :	noise $_{\text{ruido}}$
187 :	right $_{\text{bien}}$
188 :	is $_{\text{vale}}$
189 :	want $_{\text{quisiera}}$
190 :	too $_{\text{demasiado}}$
191 :	price $_{\text{precio}}$
192 :	booked $_{\text{reservé}}$
193 :	ask $_{\text{pídame}}$ ask $_{\text{pídanos}}$
194 :	night $_{\text{noche}}$
195 :	it $_{\text{tiene}}$
196 :	see $_{\text{ver}}$
197 :	quiet $_{\text{tranquilas}}$
198 :	looking $_{\text{hacerse}}$
199 :	form $_{\text{registro}}$
200 :	leave $_{\text{irnos}}$ leave $_{\text{marcharnos}}$
201 :	explain $_{\text{detallar}}$ explain $_{\text{detallarme}}$ explain $_{\text{detallarnos}}$ explain $_{\text{explicar}}$ explain $_{\text{explicarme}}$ explain $_{\text{explicarnos}}$ prepare $_{\text{preparar}}$

continúa en la página siguiente ✑

➪ *viene de la página anterior*

	prepare $_{\text{prepararme}}$ prepare $_{\text{prepararnos}}$ check $_{\text{repasar}}$ check $_{\text{repasarme}}$ check $_{\text{repasarnos}}$
202 :	change $_{\text{cambiarnos}}$
203 :	good $_{\text{buenos}}$
204 :	send $_{\text{bájeme}}$ carry $_{\text{lléveme}}$ send $_{\text{lléveme}}$ send $_{\text{llévenme}}$ put $_{\text{súbame}}$
205 :	sending $_{\text{subir}}$ sending $_{\text{subirme}}$ sending $_{\text{subirnos}}$
206 :	phone $_{\text{anotada}}$ phone $_{\text{apuntada}}$ phone $_{\text{incluida}}$
207 :	ask $_{\text{pida}}$
208 :	moving $_{\text{cambiarnos}}$
209 :	make $_{\text{hacer}}$ make $_{\text{hacerme}}$ make $_{\text{hacernos}}$
210 :	sign $_{\text{firmar}}$
211 :	credit $_{\text{tarjetas}}$
212 :	make $_{\text{haga}}$
213 :	changing $_{\text{cambiarme}}$
214 :	taking $_{\text{echáramos}}$
215 :	good $_{\text{buenas}}$
216 :	cash $_{\text{efectivo}}$
217 :	move $_{\text{cambiarnos}}$
218 :	change $_{\text{cambiarme}}$
219 :	does $_{\text{tiene}}$
220 :	night $_{\text{noches}}$ afternoon $_{\text{tardes}}$
221 :	traveler $_{\text{cheques}}$
222 :	two $_{\text{veintiuno}}$
223 :	carry $_{\text{lleva}}$ send $_{\text{lleva}}$ put $_{\text{sube}}$
224 :	very $_{\text{muchas}}$
225 :	want $_{\text{quisiéramos}}$
226 :	up $_{\text{súbame}}$
227 :	it $_{\text{hora}}$
228 :	down $_{\text{baja}}$
229 :	oh $_{\text{veinte}}$ five $_{\text{veinticinco}}$ four $_{\text{veinticuatro}}$ nine $_{\text{veintinueve}}$ eight $_{\text{veintiocho}}$ six $_{\text{veintiséis}}$ seven $_{\text{veintisiete}}$ three $_{\text{veintitrés}}$
230 :	tv $_{\text{tele}}$
231 :	they $_{\text{son}}$

continúa en la página siguiente ➪

✑ viene de la página anterior

232 :	we $_{\text{tenenos}}$
233 :	meals $_{\text{comidas}}$
234 :	care $_{\text{encargarse}}$
235 :	April $_{\text{abril}}$ August $_{\text{agosto}}$ December $_{\text{diciembre}}$ January $_{\text{enero}}$ February $_{\text{febrero}}$ July $_{\text{julio}}$ June $_{\text{junio}}$ March $_{\text{marzo}}$ May $_{\text{mayo}}$ November $_{\text{noviembre}}$ October $_{\text{octubre}}$ September $_{\text{septiembre}}$
236 :	bye $_{\text{adiós}}$
237 :	service $_{\text{servicio}}$
238 :	which $_{\text{cuál}}$
239 :	twenty-six $_{\text{veintiséis}}$
240 :	single $_{\text{sencilla}}$
241 :	accept $_{\text{aceptan}}$ accept $_{\text{aceptaría}}$ accept $_{\text{aceptarían}}$
242 :	pardon $_{\text{cómo}}$ when $_{\text{cuándo}}$ where $_{\text{dónde}}$ pardon $_{\text{dice}}$ why $_{\text{qué}}$ who $_{\text{quién}}$
243 :	ask $_{\text{pide}}$
244 :	moving $_{\text{cambiarme}}$
245 :	how $_{\text{cuántas}}$ how $_{\text{cuántos}}$
246 :	send $_{\text{bájenos}}$ send $_{\text{llévennos}}$ carry $_{\text{llévenos}}$ send $_{\text{llévenos}}$ put $_{\text{súbanos}}$
247 :	forest $_{\text{bosque}}$ lake $_{\text{lago}}$ sea $_{\text{mar}}$ river $_{\text{río}}$
248 :	Mr $_{\text{señor}}$
249 :	carry $_{\text{llevar}}$ carry $_{\text{llevarme}}$ send $_{\text{llevarme}}$ put $_{\text{subirme}}$
250 :	not $_{\text{nada}}$
251 :	phone $_{\text{anotado}}$ phone $_{\text{apuntado}}$ phone $_{\text{incluido}}$
252 :	make $_{\text{hace}}$
253 :	our $_{\text{ustedes}}$
254 :	oh $_{\text{cero}}$
255 :	eight $_{\text{dieciocho}}$ one $_{\text{once}}$
256 :	explain $_{\text{detálleme}}$ explain $_{\text{detállenos}}$ explain $_{\text{explíqueme}}$ explain $_{\text{explíquenos}}$ prepare $_{\text{prepáreme}}$ prepare $_{\text{prepárenos}}$ check $_{\text{repáseme}}$ check $_{\text{repásenos}}$
257 :	send $_{\text{subieran}}$ send $_{\text{subiese}}$ send $_{\text{subiesen}}$
258 :	sorry $_{\text{siento}}$
259 :	leave $_{\text{iré}}$ leave $_{\text{marcharé}}$ leave $_{\text{marcho}}$ leave $_{\text{me}}$
260 :	made $_{\text{hicimos}}$

continúa en la página siguiente ✑

➪ viene de la página anterior

261 :	Óscar $_{\text{Óscar}}$ Amparo $_{\text{Amparo}}$ Asunción $_{\text{Asunción}}$ Carlos $_{\text{Carlos}}$ Carmelo $_{\text{Carmelo}}$ Dulce $_{\text{Dulce}}$ Elvira $_{\text{Elvira}}$ Enrique $_{\text{Enrique}}$ Francisco $_{\text{Francisco}}$ Jaime $_{\text{Jaime}}$ José $_{\text{José}}$ María $_{\text{María}}$ Modesto $_{\text{Modesto}}$ Ricardo $_{\text{Ricardo}}$ Roberto $_{\text{Roberto}}$ Rosa $_{\text{Rosa}}$ Silvia $_{\text{Silvia}}$ Vicenta $_{\text{Vicenta}}$ Vicente $_{\text{Vicente}}$
262 :	Bacete $_{\text{Bacete}}$ Barrachina $_{\text{Barrachina}}$ Castillo $_{\text{Castillo}}$ González $_{\text{González}}$ Gracia $_{\text{Gracia}}$ Granada $_{\text{Granada}}$ Llopis $_{\text{Llopis}}$ Más $_{\text{Más}}$ Monsonís $_{\text{Monsonís}}$ Nadal $_{\text{Nadal}}$ Navarro $_{\text{Navarro}}$ Ortega $_{\text{Ortega}}$ Puig $_{\text{Puig}}$ Ruiz $_{\text{Ruiz}}$ Salgado $_{\text{Salgado}}$ Salt $_{\text{Salt}}$ Sanz $_{\text{Sanz}}$ Soler $_{\text{Soler}}$ Soria $_{\text{Soria}}$ Torres $_{\text{Torres}}$
263 :	would $_{\text{querrían}}$
264 :	lot $_{\text{mucho}}$
265 :	call $_{\text{llamarme}}$
266 :	included $_{\text{incluido}}$
267 :	Amador $_{\text{Amador}}$ Antonio $_{\text{Antonio}}$ Arturo $_{\text{Arturo}}$ Fidel $_{\text{Fidel}}$ Gerardo $_{\text{Gerardo}}$ Marta $_{\text{Marta}}$ Miguel $_{\text{Miguel}}$ Pablo $_{\text{Pablo}}$ Paloma $_{\text{Paloma}}$ Pedro $_{\text{Pedro}}$ Salvador $_{\text{Salvador}}$ Víctor $_{\text{Víctor}}$
268 :	i $_{\text{lo}}$
269 :	expensive $_{\text{cara}}$ expensive $_{\text{caro}}$

Cuadro A.1: Agrupamiento del corpus EuTrans I utilizando el algoritmo incremental dejando uno fuera. El número de clases se ha determinado de forma automática.

A.3. Colocaciones encontradas en EuTrans I

El siguiente cuadro muestra las colocaciones encontradas de forma automática en el corpus EuTrans I utilizando el algoritmo presentado en el capítulo 5. Como método de alineamiento se ha utilizado el modelo IBMm4 con 200 clases en ambos idiomas.

adiós ← good_bye	al ← in_the
al ← to_the	aseo ← a_bath
autobús ← the_bus	bájeme ← send_down
bájenme ← send_down	bájennos ← send_down
bájenos ← send_down	bajar ← send_down
bajar ← sending_down	bajara ← to_send
bajara ← to_send_down	bajarme ← send_down
bajarme ← sending_down	bajarnos ← send_down

continúa en la página siguiente ➪

➪ viene de la página anterior

bajarnos ← sending_down
bajase ← to_send_down
baje ← send_down
baje ← to_send_down
bajen ← to_send
baño ← a_bath
bosque ← of_the_forest
buenos ← good_morning
cambiarme ← change_my
cambiarme ← move_me
cambiarme ← to_change
cambiarnos ← change_our
cambiarnos ← move_us
cambiarnos ← to_change
catorce ← one_four
ciento ← number_one
cincuenta ← five_oh
cuál ← which_one
cuántas ← how_many
cuántos ← how_many
cuarto ← quarter_past
cuatrocientos ← number_four
cuatrocientos ← number_four_oh_oh
cuenta ← bill_for
déme ← give_me
darme ← give_me
darnos ← give_us
den ← to_give
dennos ← give_us
desearía ← I_would_like
desearíamos ← we_would_like
deseo ← I_want
despertaran ← to_wake
despertarme ← wake_me
despertarme ← waking_me_up
despertarnos ← waking_us_up
despertasen ← to_wake
despiérterme ← wake_me_up
bajase ← to_send
bajasen ← to_send
baje ← to_send
bajen ← send_down
bajen ← to_send_down
bien ← all_right
bosque ← the_forest
cambiar ← to_change
cambiarme ← changing_my
cambiarme ← moving_me
cambiarme ← to_move
cambiarnos ← changing_our
cambiarnos ← moving_us
cambiarnos ← to_move
cheques ← accept_traveler
ciento ← one_oh
creo ← think_that
cuánta ← how_much
cuánto ← how_much
cuarenta ← four_oh
cuarto ← to_one
cuatrocientos ← number_four_oh
cuatrocientos ← oh_oh
dé ← to_give
dénos ← give_us
darme ← giving_me
darnos ← giving_us
denme ← give_me
deseamos ← we_want
desearía ← would_like
desearíamos ← would_like
despertara ← to_wake
despertarme ← me_up
despertarme ← wake_me_up
despertarnos ← waking_us
despertase ← to_wake
despiérterme ← me_up
despiértenme ← me_up

continúa en la página siguiente ➪

viene de la página anterior

despiértenme ← wake_me_up
despiértennos ← wake_us_up
despiértenos ← wake_us_up
despierten ← to_wake
detálleme ← could_you_explain
detállenos ← could_you_explain
detalla ← could_you_explain
dieciocho ← one_eight
diecisiete ← one_seven
dieran ← to_give
diesen ← to_give
doce ← one_two
doscientos ← number_two_oh
echáramos ← us_taking
encargarse ← care_of
encargarse ← take_care_of
equivocado ← been_made
es ← it_is
existe ← there_is
explíqueme ← could_you_explain
explíquenos ← could_you_explain
gracias ← thank_you
gustaría ← we_would_like
hágame ← could_you
hágame ← make_out
háganos ← could_you_make_out
habitaciones ← have_any
hace ← make_out
hacerme ← make_out
haga ← make_out
hay ← is_there
he ← I_have
hice ← I_made
importa ← not_matter
iremos ← are_leaving
irnos ← should_leave
lago ← of_the_lake
llámeme ← could_you
despiértennos ← us_up
despiértenos ← us_up
despierte ← to_wake
detálleme ← could_you
detállenos ← could_you
detalla ← could_you
diecinueve ← one_nine
dieciséis ← one_six
diera ← to_give
diese ← to_give
diez ← one_oh
doscientos ← number_two
ducha ← a_shower
echara ← me_taking
encargarse ← care_of_the
encargarse ← taking_care_of_the
error ← an_error
estoy ← do_not
explíqueme ← could_you
explíquenos ← could_you
factura ← bill_for
gustaría ← I_would_like
gustaría ← would_like
hágame ← could_you_make_out
háganos ← could_you
háganos ← make_out
hace ← it_is
hacer ← make_out
hacernos ← make_out
hay ← are_there
hay ← there_is
hemos ← we_have
hicimos ← we_made
iré ← I_leave
iremos ← we_are_leaving
lago ← of_the
libres ← rooms_available
llámeme ← could_you_call

continúa en la página siguiente

viene de la página anterior

llámenos ← could_you
llamo ← my_name
llevara ← to_send
llevase ← to_send
lleve ← to_send
mar ← of_the
marchamos ← we_are
marcharnos ← should_leave
media ← half_past
mostrarnos ← showing_us
muchas ← thank_you_very
nada ← at_all
novecientos ← nine_oh
ochenta ← eight_oh
ochocientos ← eight_oh_oh
once ← one_one
pídame ← you_ask
pídanos ← ask_for
pídanos ← you_ask_for
parece ← think_that
pasado ← the_day_after
pedirme ← ask_for
pensión ← half_board
pide ← could_you
pienso ← think_that
podría ← could_I
precio ← price_for
prepáreme ← could_you
prepárenos ← could_you
produjo ← there_was
qué ← why_not
querría ← I_would
querríamos ← we_would_like
quiero ← I_want
quinientos ← five_oh
quisiéramos ← we_would_like
quisiera ← I_want
quisiera ← I_would_like
llámenos ← could_you_call
llamo ← my_name_is
llevaran ← to_send
llevasen ← to_send
lleven ← to_send
mar ← of_the_sea
marchamos ← we_are_leaving
marcho ← I_leave
mostrarme ← showing_me
muchas ← thank_you
mucho ← lot_of
nada ← not_at_all
noventa ← nine_oh
ochocientos ← eight_oh
ochocientos ← oh_oh
pídame ← could_you_ask_for
pídame ← you_ask_for
pídanos ← could_you_ask_for
pagar ← pay_by
pasado ← day_after
pedir ← ask_for
pedirnos ← ask_for
pide ← ask_for
pide ← could_you_ask_for
podemos ← can_we
podríamos ← could_we
precio ← the_price_for
prepáreme ← could_you_prepare
prepárenos ← could_you_prepare
puedo ← can_I
queremos ← we_want
querría ← I_would_like
querríamos ← would_like
quince ← one_five
quisiéramos ← we_want
quisiéramos ← would_like
quisiera ← I_would
río ← of_the

continúa en la página siguiente

➪ viene de la página anterior

río	← of_the_river	rellenar	← fill_in
repáseme	← could_you	repáseme	← could_you_check
repásenos	← could_you	repásenos	← could_you_check
reservé	← I_booked	reservamos	← we_booked
reservar	← to_book	ruido	← much_noise
súbame	← send_up	súbanme	← send_up
súbannos	← send_up	súbanos	← send_up
se	← a_mistake	seiscientos	← six_oh
servicio	← room_service	sesenta	← six_oh
setecientos	← seven_oh	setenta	← seven_oh
siento	← am_sorry	son	← they_are
suba	← send_up	suba	← to_send_up
suban	← send_up	suban	← to_send_up
subiera	← send_up	subiera	← to_send_up
subieran	← send_up	subieran	← to_send_up
subiese	← send_up	subiese	← to_send_up
subiesen	← send_up	subiesen	← to_send_up
subir	← sending_up	subirme	← send_up
subirme	← sending_up	subirnos	← send_up
subirnos	← sending_up	tarjetas	← accept_credit
teléfono	← a_telephone	tele	← a_tv
televisión	← a_tv	tenemos	← should_we
tenemos	← we_have	tenenos	← we_have
tengo	← I_have	tiene	← does_it
tiene	← does_it_have	tienen	← do_they_have
tienen	← they_have	todo	← everything_included
trece	← one_three	treinta	← three_oh
trescientos	← three_oh	trescientos	← three_oh_oh
vamos	← we_are	vamos	← we_are_leaving
veinte	← two_oh	veinticinco	← two_five
veinticuatro	← two_four	veintinueve	← two_nine
veintiocho	← two_eight	veintiséis	← two_six
veintisiete	← two_seven	veintitrés	← two_three
veintiuno	← two_one	vistas	← a_view
vistazo	← look_at	voy	← I_am_leaving
voy	← I_leave	voy	← am_leaving

Cuadro A.2: Colocaciones identificadas de forma automática en el corpus EuTrans I utilizando el algoritmo descrito en el capítulo 5. Como método de alineamiento se ha utilizado el modelo IBMm4 con 200 clases en ambos idiomas.

A.4. Agrupamiento de EuTrans I con colocaciones

El siguiente cuadro muestra el agrupamiento obtenido sobre el corpus EuTrans I utilizando el algoritmo incremental dejando uno fuera (ver sección 3.6). Antes de realizar el agrupamiento se utilizó el algoritmo propuesto para la identificación de colocaciones de forma automática en el idioma destino del corpus EuTrans I (ver capítulo 5). Las colocaciones detectadas se muestran en la sección anterior. El número de clases utilizado para el agrupamiento corresponde al número de clases óptimo determinado de forma automática por dicho algoritmo. Como método de alineamiento se ha utilizado el modelo IBMm4.

En la columna de la izquierda se muestra el número con el que se ha etiquetado cada una de las clases obtenidas. En la columna de la derecha se muestran las palabras extendidas que forman cada clase. En este caso, las palabras extendidas están formadas por una palabra en inglés o por una secuencia de palabras en inglés etiquetadas en cualquier caso con una palabra en castellano.

1 :	would $_{\text{aceptaría}}$ would $_{\text{aceptarían}}$ would $_{\text{haría}}$ would $_{\text{les}}$ can $_{\text{podría}}$ could $_{\text{podría}}$ could $_{\text{podrían}}$ can $_{\text{puede}}$ can $_{\text{pueden}}$ would $_{\text{querría}}$ would $_{\text{querrían}}$ will $_{\text{quiere}}$ will $_{\text{quieren}}$ do $_{\text{tiene}}$
2 :	? $_{?}$
3 :	five $_{\text{cinco}}$ four $_{\text{cuatro}}$ nine $_{\text{nueve}}$ eight $_{\text{ocho}}$ six $_{\text{seis}}$ seven $_{\text{siete}}$ three $_{\text{tres}}$
4 :	in␣the $_{\text{al}}$ to␣the $_{\text{al}}$
5 :	Álvarez $_{\text{Álvarez}}$ Aguilera $_{\text{Aguilera}}$ Alted $_{\text{Alted}}$ Aragón $_{\text{Aragón}}$ Arenas $_{\text{Arenas}}$ Arnau $_{\text{Arnau}}$ Arroyo $_{\text{Arroyo}}$ Balaguer $_{\text{Balaguer}}$ Ballester $_{\text{Ballester}}$ Barberá $_{\text{Barberá}}$ Barber $_{\text{Barber}}$ Belenguer $_{\text{Belenguer}}$ Bellver $_{\text{Bellver}}$ Beltrán $_{\text{Beltrán}}$ Berrueco $_{\text{Berrueco}}$ Betoret $_{\text{Betoret}}$ Bordons $_{\text{Bordons}}$ Borillo $_{\text{Borillo}}$ Borrás $_{\text{Borrás}}$ Botella $_{\text{Botella}}$ Cabedo $_{\text{Cabedo}}$ Cabo $_{\text{Cabo}}$ Cabrera $_{\text{Cabrera}}$ Calatayud $_{\text{Calatayud}}$ Calleja $_{\text{Calleja}}$ Campo $_{\text{Campo}}$ Cantero $_{\text{Cantero}}$ Carpio $_{\text{Carpio}}$ Cerezo $_{\text{Cerezo}}$ Colomer $_{\text{Colomer}}$ Cornelles $_{\text{Cornelles}}$ Cutillas $_{\text{Cutillas}}$ Díaz $_{\text{Díaz}}$ Edo $_{\text{Edo}}$ Escrig $_{\text{Escrig}}$ Espinosa $_{\text{Espinosa}}$ Fernández $_{\text{Fernández}}$ Ferrando $_{\text{Ferrando}}$ Fuster $_{\text{Fuster}}$ Gómez $_{\text{Gómez}}$ Gallego $_{\text{Gallego}}$ García $_{\text{García}}$ Gil $_{\text{Gil}}$ Gimeno $_{\text{Gimeno}}$ Gual $_{\text{Gual}}$ Guardiola $_{\text{Guardiola}}$ Guijarro $_{\text{Guijarro}}$ Gumbau $_{\text{Gumbau}}$ Herrero $_{\text{Herrero}}$ Ibáñez $_{\text{Ibáñez}}$ Iborra $_{\text{Iborra}}$ Ivars $_{\text{Ivars}}$ Izquierdo $_{\text{Izquierdo}}$ Jiménez $_{\text{Jiménez}}$ López $_{\text{López}}$ Llorens $_{\text{Llorens}}$ Lobo $_{\text{Lobo}}$ Maestro $_{\text{Maestro}}$ Marqués $_{\text{Marqués}}$ Martí $_{\text{Martí}}$ Martínez $_{\text{Martínez}}$ Meda $_{\text{Meda}}$ Mestre $_{\text{Mestre}}$ Mingot $_{\text{Mingot}}$ Mira $_{\text{Mira}}$ Miralles $_{\text{Miralles}}$ Moliner $_{\text{Moliner}}$ Monferrer $_{\text{Monferrer}}$ Montero $_{\text{Montero}}$ Montoro $_{\text{Montoro}}$ Morales $_{\text{Morales}}$ Moreno $_{\text{Moreno}}$ Moya $_{\text{Moya}}$ Nebot $_{\text{Nebot}}$

continúa en la página siguiente ➪

➪ *viene de la página anterior*

Orenga $_{\text{Orenga}}$ Ortiz $_{\text{Ortiz}}$ Pérez $_{\text{Pérez}}$ Paches $_{\text{Paches}}$ Padilla $_{\text{Padilla}}$
Pallarés $_{\text{Pallarés}}$ Pastor $_{\text{Pastor}}$ Peinado $_{\text{Peinado}}$ Peris $_{\text{Peris}}$
Peñarroja $_{\text{Peñarroja}}$ Piquer $_{\text{Piquer}}$ Pitarch $_{\text{Pitarch}}$ Quereda $_{\text{Quereda}}$
Querol $_{\text{Querol}}$ Ríos $_{\text{Ríos}}$ Ródenas $_{\text{Ródenas}}$ Ramírez $_{\text{Ramírez}}$ Ramos $_{\text{Ramos}}$
Redondo $_{\text{Redondo}}$ Revilla $_{\text{Revilla}}$ Rivera $_{\text{Rivera}}$ Roda $_{\text{Roda}}$
Rodríguez $_{\text{Rodríguez}}$ Romero $_{\text{Romero}}$ Royo $_{\text{Royo}}$ Rubio $_{\text{Rubio}}$ Sáez $_{\text{Sáez}}$
Salsas $_{\text{Salsas}}$ Sanfeliú $_{\text{Sanfeliú}}$ Santos $_{\text{Santos}}$ Segura $_{\text{Segura}}$
Serrano $_{\text{Serrano}}$ Sevilla $_{\text{Sevilla}}$ Silvestre $_{\text{Silvestre}}$ Suárez $_{\text{Suárez}}$
Tena $_{\text{Tena}}$ Vázquez $_{\text{Vázquez}}$ Valls $_{\text{Valls}}$ Varela $_{\text{Varela}}$ Velasco $_{\text{Velasco}}$
Viciano $_{\text{Viciano}}$ Vidal $_{\text{Vidal}}$ Vilanova $_{\text{Vilanova}}$ Villanueva $_{\text{Villanueva}}$

6 : my $_{\text{los}}$ our $_{\text{los}}$ our $_{\text{nuestros}}$

7 : fourteenth $_{\text{catorce}}$ fifth $_{\text{cinco}}$ fourth $_{\text{cuatro}}$ nineteenth $_{\text{diecinueve}}$
eighteenth $_{\text{dieciocho}}$ sixteenth $_{\text{dieciséis}}$ seventeenth $_{\text{diecisiete}}$
tenth $_{\text{diez}}$ twelfth $_{\text{doce}}$ second $_{\text{dos}}$ ninth $_{\text{nueve}}$ eighth $_{\text{ocho}}$ eleventh $_{\text{once}}$
fifteenth $_{\text{quince}}$ sixth $_{\text{seis}}$ seventh $_{\text{siete}}$ thirteenth $_{\text{trece}}$
thirtieth $_{\text{treinta}}$ thirty-first $_{\text{treinta}}$ third $_{\text{tres}}$ first $_{\text{uno}}$
twentieth $_{\text{veinte}}$ twenty-fifth $_{\text{veinticinco}}$ twenty-fourth $_{\text{veinticuatro}}$
twenty-second $_{\text{veintidós}}$ twenty-ninth $_{\text{veintinueve}}$
twenty-eighth $_{\text{veintiocho}}$ twenty-sixth $_{\text{veintiséis}}$
twenty-seventh $_{\text{veintisiete}}$ twenty-third $_{\text{veintitrés}}$
twenty-first $_{\text{veintiuno}}$

8 : a␣bath $_{\text{aseo}}$ a␣bath $_{\text{baño}}$ a␣shower $_{\text{ducha}}$ a␣telephone $_{\text{teléfono}}$
a␣tv $_{\text{televisión}}$

9 : Sunday $_{\text{domingo}}$ Thursday $_{\text{jueves}}$ Monday $_{\text{lunes}}$ Tuesday $_{\text{martes}}$
Wednesday $_{\text{miércoles}}$ Saturday $_{\text{sábado}}$ Friday $_{\text{viernes}}$

10 :
Ángel $_{\text{Ángel}}$ Alicia $_{\text{Alicia}}$ Amelia $_{\text{Amelia}}$ Ana $_{\text{Ana}}$ Andrés $_{\text{Andrés}}$
Beatriz $_{\text{Beatriz}}$ César $_{\text{César}}$ Carmen $_{\text{Carmen}}$ Celestino $_{\text{Celestino}}$
Concepción $_{\text{Concepción}}$ Cristina $_{\text{Cristina}}$ Daniel $_{\text{Daniel}}$ Eduardo $_{\text{Eduardo}}$
Emilio $_{\text{Emilio}}$ Eva $_{\text{Eva}}$ Federico $_{\text{Federico}}$ Gloria $_{\text{Gloria}}$ Gregorio $_{\text{Gregorio}}$
Ignacio $_{\text{Ignacio}}$ Inmaculada $_{\text{Inmaculada}}$ Isabel $_{\text{Isabel}}$ Jacinto $_{\text{Jacinto}}$
Javier $_{\text{Javier}}$ Jesús $_{\text{Jesús}}$ Joaquín $_{\text{Joaquín}}$ Jorge $_{\text{Jorge}}$ Juan $_{\text{Juan}}$
Julia $_{\text{Julia}}$ Julio $_{\text{Julio}}$ Lidia $_{\text{Lidia}}$ Luis $_{\text{Luis}}$ Manuel $_{\text{Manuel}}$
Manuela $_{\text{Manuela}}$ Margarita $_{\text{Margarita}}$ Marina $_{\text{Marina}}$ Micaela $_{\text{Micaela}}$
Pilar $_{\text{Pilar}}$ Rafael $_{\text{Rafael}}$ Rosalía $_{\text{Rosalía}}$ Rosario $_{\text{Rosario}}$
Santiago $_{\text{Santiago}}$ Sergio $_{\text{Sergio}}$ Sonia $_{\text{Sonia}}$ Susana $_{\text{Susana}}$ Teresa $_{\text{Teresa}}$
Tomás $_{\text{Tomás}}$ Victoria $_{\text{Victoria}}$ Virginia $_{\text{Virginia}}$

11 : double $_{\text{doble}}$ single $_{\text{individual}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

12 :	room $_{\text{habitación}}$
13 :	fourteen $_{\text{catorce}}$ nineteen $_{\text{diecinueve}}$ eighteen $_{\text{dieciocho}}$ sixteen $_{\text{dieciséis}}$ seventeen $_{\text{diecisiete}}$ fifteen $_{\text{quince}}$ thirteen $_{\text{trece}}$ thirty $_{\text{treinta}}$ twenty $_{\text{veinte}}$ twenty-five $_{\text{veinticinco}}$ twenty-four $_{\text{veinticuatro}}$ twenty-two $_{\text{veintidós}}$ twenty-nine $_{\text{veintinueve}}$ twenty-eight $_{\text{veintiocho}}$ twenty-six $_{\text{veintiséis}}$ twenty-seven $_{\text{veintisiete}}$ twenty-three $_{\text{veintitrés}}$ twenty-one $_{\text{veintiuno}}$
14 :	changing␣my $_{\text{cambiarme}}$ changing␣our $_{\text{cambiarnos}}$
15 :	minibar $_{\text{bar}}$ bathroom $_{\text{baño}}$ safe $_{\text{fuerte}}$
16 :	leave $_{\text{ir}}$ leave $_{\text{marchar}}$
17 :	with $_{\text{tenga}}$
18 :	to␣give $_{\text{dé}}$ to␣give $_{\text{den}}$ to␣wake $_{\text{despertara}}$ to␣wake $_{\text{despertaran}}$ to␣wake $_{\text{despertase}}$ to␣wake $_{\text{despertasen}}$ to␣wake $_{\text{despierte}}$ to␣wake $_{\text{despierten}}$ to␣give $_{\text{diera}}$ to␣give $_{\text{dieran}}$ to␣give $_{\text{diese}}$ to␣give $_{\text{diesen}}$
19 :	today $_{\text{hoy}}$
20 :	. $_{.}$
21 :	hot $_{\text{calor}}$ cold $_{\text{frío}}$
22 :	call $_{\text{llamar}}$ call $_{\text{llamarme}}$ ask␣for $_{\text{pedir}}$ ask␣for $_{\text{pedirme}}$
23 :	sending␣down $_{\text{bajar}}$ sending␣down $_{\text{bajarme}}$ sending␣down $_{\text{bajarnos}}$ sending $_{\text{llevar}}$ sending $_{\text{llevarme}}$ sending␣up $_{\text{subir}}$ sending␣up $_{\text{subirme}}$ sending␣up $_{\text{subirnos}}$
24 :	at $_{\text{las}}$
25 :	my $_{\text{mi}}$
26 :	morning $_{\text{mañana}}$ evening $_{\text{noche}}$ afternoon $_{\text{tarde}}$
27 :	I␣would␣like $_{\text{desearía}}$ I␣want $_{\text{deseo}}$ I␣want $_{\text{quiero}}$
28 :	send␣down $_{\text{baje}}$ carry $_{\text{lleve}}$ send $_{\text{lleve}}$ send $_{\text{lleven}}$ put $_{\text{suba}}$ send␣up $_{\text{suba}}$ send␣up $_{\text{suban}}$
29 :	in $_{\text{en}}$
30 :	quarter␣past $_{\text{cuarto}}$
31 :	keys $_{\text{llaves}}$
32 :	number␣one $_{\text{ciento}}$ number␣four $_{\text{cuatrocientos}}$ number␣two $_{\text{doscientos}}$
33 :	bus $_{\text{autobús}}$ car $_{\text{coche}}$
34 :	double $_{\text{dobles}}$ single $_{\text{individuales}}$

continúa en la página siguiente ➪

viene de la página anterior

35 :	quiet $_{\text{tranquila}}$
36 :	bill_for $_{\text{cuenta}}$ bill_for $_{\text{factura}}$
37 :	wake_me_up $_{\text{despiérteme}}$ wake_me_up $_{\text{despiértenme}}$ wake_us_up $_{\text{despiértennos}}$ wake_us_up $_{\text{despiértenos}}$
38 :	there_is $_{\text{existe}}$ there_was $_{\text{produjo}}$
39 :	cards $_{\text{crédito}}$ checks $_{\text{viaje}}$
40 :	the $_{\text{la}}$
41 :	town $_{\text{ciudad}}$ mountain $_{\text{montaña}}$
42 :	you $_{\text{¿}}$
43 :	for $_{\text{de}}$
44 :	waking_me_up $_{\text{despertarme}}$ waking_us_up $_{\text{despertarnos}}$
45 :	to $_{\text{de}}$
46 :	the $_{\text{las}}$
47 :	taxi $_{\text{taxi}}$
48 :	reservation $_{\text{reserva}}$
49 :	for $_{\text{para}}$
50 :	my $_{\text{la}}$ our $_{\text{la}}$
51 :	to $_{\text{a}}$
52 :	ten $_{\text{diez}}$ twelve $_{\text{doce}}$ eleven $_{\text{once}}$
53 :	key $_{\text{llave}}$
54 :	and $_{\text{y}}$
55 :	half_past $_{\text{media}}$
56 :	pardon $_{\text{cómo}}$ which_one $_{\text{cuál}}$ when $_{\text{cuándo}}$ how_much $_{\text{cuánta}}$ how_many $_{\text{cuántas}}$ how_many $_{\text{cuántos}}$ where $_{\text{dónde}}$ pardon $_{\text{dice}}$ why $_{\text{qué}}$ why_not $_{\text{qué}}$ who $_{\text{quién}}$
57 :	showing_me $_{\text{mostrarme}}$ showing_us $_{\text{mostrarnos}}$
58 :	made $_{\text{hecha}}$
59 :	to_change $_{\text{cambiar}}$ to_change $_{\text{cambiarme}}$
60 :	what $_{\text{qué}}$
61 :	has $_{\text{ha}}$
62 :	rooms $_{\text{habitaciones}}$
63 :	one_four $_{\text{catorce}}$ five_oh $_{\text{cincuenta}}$ four_oh $_{\text{cuarenta}}$ one_nine $_{\text{diecinueve}}$ one_eight $_{\text{dieciocho}}$ one_six $_{\text{dieciséis}}$ one_seven $_{\text{diecisiete}}$ one_oh $_{\text{diez}}$

continúa en la página siguiente

➪ viene de la página anterior

	one_two $_{\text{doce}}$ nine_oh $_{\text{noventa}}$ eight_oh $_{\text{ochenta}}$ one_one $_{\text{once}}$ one_five $_{\text{quince}}$ six_oh $_{\text{sesenta}}$ seven_oh $_{\text{setenta}}$ one_three $_{\text{trece}}$ three_oh $_{\text{treinta}}$ two_oh $_{\text{veinte}}$ two_five $_{\text{veinticinco}}$ two_four $_{\text{veinticuatro}}$ two_nine $_{\text{veintinueve}}$ two_eight $_{\text{veintiocho}}$ two_six $_{\text{veintiséis}}$ two_seven $_{\text{veintisiete}}$ two_three $_{\text{veintitrés}}$ two_one $_{\text{veintiuno}}$
64 :	two $_{\text{dos}}$
65 :	suitcases $_{\text{maletas}}$
66 :	to_send_down $_{\text{bajara}}$ to_send_down $_{\text{bajase}}$ to_send_down $_{\text{baje}}$ to_send_down $_{\text{bajen}}$ to_send $_{\text{llevara}}$ to_send $_{\text{llevaran}}$ to_send $_{\text{llevase}}$ to_send $_{\text{llevasen}}$ to_send $_{\text{lleve}}$ to_send $_{\text{lleven}}$ to_send_up $_{\text{suba}}$ to_send_up $_{\text{suban}}$ to_send_up $_{\text{subiera}}$ to_send_up $_{\text{subieran}}$ to_send_up $_{\text{subiese}}$ to_send_up $_{\text{subiesen}}$
67 :	on $_{\text{el}}$
68 :	give $_{\text{dar}}$ wake $_{\text{despertar}}$
69 :	I_leave $_{\text{iré}}$ we_are_leaving $_{\text{iremos}}$ we_are_leaving $_{\text{marchamos}}$ I_leave $_{\text{marcho}}$ we_are_leaving $_{\text{vamos}}$ I_am_leaving $_{\text{voy}}$ I_leave $_{\text{voy}}$
70 :	are $_{\text{están}}$
71 :	a $_{\text{la}}$
72 :	! $_{\text{¡}}$
73 :	I_booked $_{\text{reservé}}$ we_booked $_{\text{reservamos}}$
74 :	sign $_{\text{firmar}}$ fill_in $_{\text{rellenar}}$
75 :	is $_{\text{es}}$
76 :	quarter $_{\text{menos}}$
77 :	explain $_{\text{detallar}}$ explain $_{\text{detallarnos}}$ explain $_{\text{explicar}}$ explain $_{\text{explicarme}}$ explain $_{\text{explicarnos}}$ make_out $_{\text{hacer}}$ make_out $_{\text{hacernos}}$ prepare $_{\text{preparar}}$ prepare $_{\text{prepararme}}$ prepare $_{\text{prepararnos}}$ check $_{\text{repasar}}$ check $_{\text{repasarme}}$ check $_{\text{repasarnos}}$
78 :	me $_{\text{me}}$ us $_{\text{nos}}$
79 :	at $_{\text{a}}$
80 :	until $_{\text{hasta}}$
81 :	tomorrow $_{\text{mañana}}$
82 :	day $_{\text{día}}$
83 :	tax $_{\text{valor}}$
84 :	it_is $_{\text{es}}$ they_are $_{\text{son}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

85 :	the $_{\text{el}}$
86 :	registration $_{\text{registro}}$
87 :	everything_included $_{\text{todo}}$
88 :	? $_{\text{.}}$
89 :	to_book $_{\text{reservar}}$
90 :	I_have $_{\text{he}}$
91 :	a $_{\text{una}}$
92 :	bag $_{\text{bolsa}}$
93 :	the $_{\text{día}}$
94 :	pay $_{\text{abonar}}$ pay $_{\text{pagar}}$
95 :	phone $_{\text{anotada}}$ phone $_{\text{apuntada}}$ phone $_{\text{incluida}}$
96 :	hot $_{\text{agua}}$ air $_{\text{aire}}$
97 :	mistake $_{\text{equivocación}}$
98 :	number_four_oh_oh $_{\text{cuatrocientos}}$ cost $_{\text{cuesta}}$
99 :	does_it_have $_{\text{tiene}}$ do_they_have $_{\text{tienen}}$
100 :	extras $_{\text{extras}}$ expenses $_{\text{gastos}}$ taxes $_{\text{impuestos}}$
101 :	could_I $_{\text{podría}}$ could_we $_{\text{podríamos}}$
102 :	conditioning $_{\text{acondicionado}}$ water $_{\text{caliente}}$
103 :	bill $_{\text{recibo}}$
104 :	luggage $_{\text{bultos}}$
105 :	o'clock $_{\text{punto}}$
106 :	luggage $_{\text{equipaje}}$
107 :	booked $_{\text{reservada}}$
108 :	Miss $_{\text{señorita}}$
109 :	an_error $_{\text{error}}$
110 :	good $_{\text{buena}}$
111 :	see $_{\text{ver}}$
112 :	look $_{\text{hacerse}}$ mind $_{\text{importaría}}$
113 :	accept_traveler $_{\text{cheques}}$ accept_credit $_{\text{tarjetas}}$
114 :	is $_{\text{está}}$
115 :	okay $_{\text{acuerdo}}$ good_bye $_{\text{adiós}}$ good_morning $_{\text{buenos}}$ okay $_{\text{conforme}}$ okay $_{\text{correcto}}$ sorry $_{\text{disculpe}}$ thank_you $_{\text{gracias}}$ hello $_{\text{hola}}$ no $_{\text{no}}$ sorry $_{\text{perdón}}$ sorry $_{\text{perdone}}$ yes $_{\text{sí}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

116 :	one $_{\text{una}}$
117 :	travel $_{\text{viaje}}$
118 :	included $_{\text{del}}$
119 :	view $_{\text{vista}}$
120 :	my $_{\text{las}}$ our $_{\text{las}}$
121 :	there $_{\text{se}}$
122 :	think␣that $_{\text{parece}}$
123 :	the␣bus $_{\text{autobús}}$
124 :	warmer $_{\text{cálida}}$ warmer $_{\text{fría}}$ quieter $_{\text{más}}$ quieter $_{\text{ruido}}$ quieter $_{\text{ruidosa}}$
125 :	look␣at $_{\text{vistazo}}$
126 :	been␣made $_{\text{equivocado}}$
127 :	time $_{\text{hora}}$
128 :	tonight $_{\text{esta}}$
129 :	wake $_{\text{despertarnos}}$
130 :	bill $_{\text{cuenta}}$ bill $_{\text{factura}}$
131 :	this $_{\text{esta}}$
132 :	number␣four␣oh $_{\text{cuatrocientos}}$ number␣two␣oh $_{\text{doscientos}}$
133 :	available $_{\text{libre}}$
134 :	another $_{\text{otra}}$
135 :	have $_{\text{libre}}$
136 :	should $_{\text{tengo}}$
137 :	hot $_{\text{calurosa}}$ expensive $_{\text{caras}}$ cold $_{\text{fría}}$ noisy $_{\text{ruidosa}}$
138 :	breakfast $_{\text{desayuno}}$ room␣service $_{\text{servicio}}$
139 :	all␣right $_{\text{bien}}$
140 :	half␣board $_{\text{pensión}}$
141 :	including $_{\text{incluyendo}}$
142 :	very $_{\text{muy}}$
143 :	reception $_{\text{recepción}}$
144 :	our $_{\text{nuestra}}$
145 :	a␣view $_{\text{vistas}}$
146 :	made $_{\text{hecho}}$
147 :	problem $_{\text{problema}}$
148 :	any $_{\text{habitaciones}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

149 :	are_there $_{\text{hay}}$
150 :	how_much $_{\text{cuánto}}$
151 :	would $_{\text{le}}$
152 :	my $_{\text{el}}$ our $_{\text{el}}$
153 :	the_day_after $_{\text{pasado}}$
154 :	should_leave $_{\text{irnos}}$ should_leave $_{\text{marcharnos}}$
155 :	move_me $_{\text{cambiarme}}$ move_us $_{\text{cambiarnos}}$
156 :	could_you_call $_{\text{llámeme}}$ could_you_ask_for $_{\text{pídame}}$
157 :	call $_{\text{llama}}$
158 :	nine $_{\text{novecientos}}$ eight $_{\text{ochocientos}}$ five $_{\text{quinientos}}$ six $_{\text{seiscientos}}$ seven $_{\text{setecientos}}$ three $_{\text{trescientos}}$
159 :	been $_{\text{producido}}$
160 :	we_want $_{\text{quisiéramos}}$ I_want $_{\text{quisiera}}$
161 :	booked $_{\text{reservadas}}$
162 :	ask $_{\text{pida}}$
163 :	to_move $_{\text{cambiarme}}$
164 :	Mrs $_{\text{señora}}$
165 :	the_price_for $_{\text{precio}}$
166 :	send $_{\text{subir}}$
167 :	one_oh $_{\text{ciento}}$ nine_oh $_{\text{novecientos}}$ eight_oh $_{\text{ochocientos}}$ five_oh $_{\text{quinientos}}$ six_oh $_{\text{seiscientos}}$ seven_oh $_{\text{setecientos}}$ three_oh $_{\text{trescientos}}$
168 :	we_have $_{\text{tenenos}}$
169 :	form $_{\text{hoja}}$
170 :	book $_{\text{reservar}}$
171 :	we_have $_{\text{tenemos}}$
172 :	week $_{\text{semana}}$
173 :	phone $_{\text{anotado}}$ phone $_{\text{apuntado}}$ phone $_{\text{incluido}}$
174 :	rooms_available $_{\text{libres}}$
175 :	have_any $_{\text{habitaciones}}$
176 :	my $_{\text{mis}}$
177 :	I_made $_{\text{hice}}$
178 :	giving_me $_{\text{darme}}$ giving_us $_{\text{darnos}}$
179 :	should_we $_{\text{tenemos}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

180 :	a $_{\text{alguna}}$
181 :	have_any $_{\text{libres}}$
182 :	days $_{\text{días}}$ nights $_{\text{noches}}$ weeks $_{\text{semanas}}$
183 :	very $_{\text{mucho}}$
184 :	bags $_{\text{bolsas}}$
185 :	send $_{\text{baja}}$ send $_{\text{bajar}}$ send $_{\text{bajarme}}$ send_down $_{\text{bajarme}}$ send $_{\text{bajarnos}}$ take_care_of $_{\text{encargarse}}$ carry $_{\text{lleva}}$ send $_{\text{lleva}}$ carry $_{\text{llevar}}$ send $_{\text{llevar}}$ carry $_{\text{llevarme}}$ send $_{\text{llevarme}}$ carry $_{\text{llevarnos}}$ send $_{\text{llevarnos}}$ put $_{\text{sube}}$ put $_{\text{subir}}$ put $_{\text{subirme}}$ send_up $_{\text{subirme}}$ put $_{\text{subirnos}}$
186 :	booked $_{\text{reservado}}$
187 :	next $_{\text{próximo}}$
188 :	night $_{\text{noches}}$ afternoon $_{\text{tardes}}$
189 :	up $_{\text{usted}}$ up $_{\text{ustedes}}$
190 :	added $_{\text{sobre}}$
191 :	five $_{\text{cincuenta}}$ four $_{\text{cuarenta}}$ nine $_{\text{noventa}}$ eight $_{\text{ochenta}}$ six $_{\text{sesenta}}$ seven $_{\text{setenta}}$ three $_{\text{treinta}}$
192 :	two $_{\text{veintidós}}$
193 :	wake_me_up $_{\text{despertarme}}$
194 :	my_name_is $_{\text{llamo}}$
195 :	by $_{\text{con}}$
196 :	we $_{\text{hemos}}$ we $_{\text{tenemos}}$
197 :	station $_{\text{estación}}$
198 :	night $_{\text{noche}}$
199 :	full $_{\text{pensión}}$
200 :	date $_{\text{estamos}}$
201 :	I_have $_{\text{tengo}}$
202 :	do $_{\text{tienen}}$
203 :	quiet $_{\text{tranquilas}}$
204 :	suitcase $_{\text{maleta}}$
205 :	we_would_like $_{\text{gustaría}}$
206 :	give_me $_{\text{déme}}$ give_us $_{\text{dénos}}$ give_me $_{\text{denme}}$ give_us $_{\text{dennos}}$
207 :	our $_{\text{nuestro}}$
208 :	too $_{\text{demasiado}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

209 :	I $_{\text{he}}$
210 :	has $_{\text{han}}$
211 :	a␣mistake $_{\text{se}}$
212 :	board $_{\text{completa}}$
213 :	give␣me $_{\text{darme}}$ give␣us $_{\text{darnos}}$
214 :	leave $_{\text{marcharé}}$
215 :	can␣we $_{\text{podemos}}$ can␣I $_{\text{puedo}}$
216 :	it␣is $_{\text{hace}}$
217 :	check $_{\text{cheques}}$
218 :	any $_{\text{libres}}$
219 :	I␣would␣like $_{\text{querría}}$ I␣would␣like $_{\text{quisiera}}$
220 :	moving␣me $_{\text{cambiarme}}$ moving␣us $_{\text{cambiarnos}}$
221 :	call $_{\text{llame}}$
222 :	could␣you␣explain $_{\text{detálleme}}$ could␣you␣explain $_{\text{detállenos}}$ could␣you␣explain $_{\text{detalla}}$ could␣you␣explain $_{\text{explíqueme}}$ could␣you␣explain $_{\text{explíquenos}}$ could␣you␣make␣out $_{\text{hágame}}$ could␣you␣prepare $_{\text{prepáreme}}$ could␣you␣check $_{\text{repáseme}}$ could␣you␣check $_{\text{repásenos}}$ does $_{\text{tiene}}$
223 :	name $_{\text{nombre}}$
224 :	send␣down $_{\text{bajar}}$ send␣down $_{\text{bajarnos}}$ send␣up $_{\text{subirnos}}$
225 :	explain $_{\text{explica}}$ make␣out $_{\text{hace}}$ prepare $_{\text{prepara}}$ check $_{\text{repasa}}$
226 :	lot␣of $_{\text{ruido}}$
227 :	taking␣care␣of␣the $_{\text{encargarse}}$ looking $_{\text{hacerse}}$
228 :	noise $_{\text{mucho}}$
229 :	agree $_{\text{acuerdo}}$
230 :	with $_{\text{con}}$
231 :	I␣would␣like $_{\text{gustaría}}$
232 :	eight␣oh␣oh $_{\text{ochocientos}}$ three␣oh␣oh $_{\text{trescientos}}$ one $_{\text{uno}}$
233 :	change␣my $_{\text{cambiarme}}$ change␣our $_{\text{cambiarnos}}$
234 :	of␣the␣forest $_{\text{bosque}}$ of␣the␣lake $_{\text{lago}}$ of␣the␣sea $_{\text{mar}}$ of␣the␣river $_{\text{río}}$
235 :	, $_{,}$
236 :	us␣taking $_{\text{echáramos}}$ me␣taking $_{\text{echara}}$
237 :	credit $_{\text{tarjetas}}$

continúa en la página siguiente ➪

➪ viene de la página anterior

238 :	send $_{\text{bájenos}}$ send_down $_{\text{bájenos}}$ send $_{\text{baje}}$ send_down $_{\text{bajen}}$ send $_{\text{llévennos}}$ carry $_{\text{llévenos}}$ send $_{\text{llévenos}}$ send_up $_{\text{súbanme}}$ send_up $_{\text{súbannos}}$ put $_{\text{súbanos}}$ send_up $_{\text{súbanos}}$
239 :	included $_{\text{incluido}}$
240 :	to_move $_{\text{cambiarnos}}$
241 :	is_there $_{\text{hay}}$
242 :	April $_{\text{abril}}$ August $_{\text{agosto}}$ December $_{\text{diciembre}}$ January $_{\text{enero}}$ February $_{\text{febrero}}$ July $_{\text{julio}}$ June $_{\text{junio}}$ March $_{\text{marzo}}$ May $_{\text{mayo}}$ November $_{\text{noviembre}}$ October $_{\text{octubre}}$ September $_{\text{septiembre}}$
243 :	we_have $_{\text{hemos}}$
244 :	a_tv $_{\text{tele}}$
245 :	do_not $_{\text{estoy}}$
246 :	our $_{\text{nuestras}}$
247 :	send_down $_{\text{bájeme}}$ send_down $_{\text{bájenme}}$ send_down $_{\text{bájennos}}$ send $_{\text{lléveme}}$ send $_{\text{llévenme}}$ put $_{\text{súbame}}$
248 :	single $_{\text{sencilla}}$
249 :	oh $_{\text{cero}}$
250 :	cash $_{\text{efectivo}}$
251 :	accept $_{\text{dinero}}$
252 :	much_noise $_{\text{ruido}}$
253 :	card $_{\text{crédito}}$
254 :	Mr $_{\text{señor}}$
255 :	there_is $_{\text{hay}}$
256 :	number $_{\text{número}}$
257 :	sending $_{\text{bajar}}$ sending $_{\text{bajarme}}$ after $_{\text{cargo}}$ carrying $_{\text{llevar}}$ carrying $_{\text{llevarme}}$ carrying $_{\text{llevarnos}}$ putting $_{\text{subir}}$ putting $_{\text{subirme}}$
258 :	expensive $_{\text{cara}}$ expensive $_{\text{caro}}$
259 :	I $_{\text{no}}$
260 :	call $_{\text{llamarnos}}$ ask_for $_{\text{pedirnos}}$
261 :	pay_by $_{\text{pagar}}$
262 :	meals $_{\text{comidas}}$
263 :	we_would_like $_{\text{querríamos}}$ we_would_like $_{\text{quisiéramos}}$
264 :	explain $_{\text{detalle}}$ explain $_{\text{explique}}$ make_out $_{\text{haga}}$ prepare $_{\text{prepare}}$ check $_{\text{repase}}$

continúa en la página siguiente ➪

➪ *viene de la página anterior*

265 :	send $_{\text{bájeme}}$ down $_{\text{bajasen}}$ carry $_{\text{lléveme}}$ send_up $_{\text{súbame}}$
266 :	leave $_{\text{irme}}$ leave $_{\text{marcharme}}$
267 :	good $_{\text{buenas}}$
268 :	not_matter $_{\text{importa}}$
269 :	Óscar $_{\text{Óscar}}$ Amparo $_{\text{Amparo}}$ Asunción $_{\text{Asunción}}$ Carlos $_{\text{Carlos}}$ Carmelo $_{\text{Carmelo}}$ Dulce $_{\text{Dulce}}$ Elvira $_{\text{Elvira}}$ Enrique $_{\text{Enrique}}$ Francisco $_{\text{Francisco}}$ Jaime $_{\text{Jaime}}$ José $_{\text{José}}$ María $_{\text{María}}$ Modesto $_{\text{Modesto}}$ Ricardo $_{\text{Ricardo}}$ Roberto $_{\text{Roberto}}$ Rosa $_{\text{Rosa}}$ Silvia $_{\text{Silvia}}$ Vicenta $_{\text{Vicenta}}$ Vicente $_{\text{Vicente}}$
270 :	we_want $_{\text{deseamos}}$ we_would_like $_{\text{desearíamos}}$ we_want $_{\text{queremos}}$
271 :	to_change $_{\text{cambiarnos}}$
272 :	Bacete $_{\text{Bacete}}$ Barrachina $_{\text{Barrachina}}$ Castillo $_{\text{Castillo}}$ González $_{\text{González}}$ Gracia $_{\text{Gracia}}$ Granada $_{\text{Granada}}$ Llopis $_{\text{Llopis}}$ Más $_{\text{Más}}$ Monsonís $_{\text{Monsonís}}$ Nadal $_{\text{Nadal}}$ Navarro $_{\text{Navarro}}$ Ortega $_{\text{Ortega}}$ Puig $_{\text{Puig}}$ Ruiz $_{\text{Ruiz}}$ Salgado $_{\text{Salgado}}$ Salt $_{\text{Salt}}$ Sanz $_{\text{Sanz}}$ Soler $_{\text{Soler}}$ Soria $_{\text{Soria}}$ Torres $_{\text{Torres}}$
273 :	could_you_make_out $_{\text{háganos}}$ could_you_prepare $_{\text{prepárenos}}$
274 :	could_you_call $_{\text{llámenos}}$ could_you_ask_for $_{\text{pídanos}}$ could_you_ask_for $_{\text{pide}}$
275 :	we_made $_{\text{hicimos}}$
276 :	not_at_all $_{\text{nada}}$
277 :	Amador $_{\text{Amador}}$ Antonio $_{\text{Antonio}}$ Arturo $_{\text{Arturo}}$ Fidel $_{\text{Fidel}}$ Gerardo $_{\text{Gerardo}}$ Marta $_{\text{Marta}}$ Miguel $_{\text{Miguel}}$ Pablo $_{\text{Pablo}}$ Paloma $_{\text{Paloma}}$ Pedro $_{\text{Pedro}}$ Salvador $_{\text{Salvador}}$ Víctor $_{\text{Víctor}}$
278 :	explain $_{\text{detallarme}}$ make_out $_{\text{hacerme}}$
279 :	sending $_{\text{bajarnos}}$ sending $_{\text{llevarnos}}$ putting $_{\text{subirnos}}$

Cuadro A.3: Agrupamiento del corpus EuTrans I con colocaciones utilizando el algoritmo incremental dejando uno fuera. El número de clases se ha determinado de forma automática.

APÉNDICE B

Resultados numéricos

B.1. Introducción

Los resultados numéricos mostrados en este apéndice corresponden a los resultados experimentales presentados en forma de gráficas en el capítulo 6. Para facilitar el paso contrario, dada una tabla de resultados localizar la gráfica correspondiente, se ha incluido al final de la descripción de cada tabla, la referencia a la gráfica en la que se muestran dichos valores.

En este capítulo sólo presentamos los resultados numéricos. Ver el capítulo 6 para una explicación más detallada de cada uno de los experimentos y nuestra interpretación de los resultados obtenidos.

Como información adicional, para los experimentos bilingües de traducción se incluyen además de los valores de BLEU y WER representados en el capítulo 6, los valores de PER y SER obtenidos.

B.2. Agrupamiento monolingüe

A modo de recordatorio, los métodos de agrupamiento monolingüe que se comparan en los siguientes experimentos son:

- `srilm_inc`. Las clases se construyen utilizando un algoritmo voraz incremental. Comienza con una clase para cada una de las C palabras más frecuentes y añade una palabra cada vez. Esta es una implementación del algoritmo de $O(V \cdot C^2)$, donde V es el tamaño del vocabulario, descrito en [BDd+92].

- `srilm_full`. Las clases se construyen utilizando un algoritmo voraz sobre todas las clases comenzando por una clase por palabra. Es una implementación del algoritmo de $O(V^3)$ descrito en [BDd+92].

- `mkcls`. El método implementado se describe en [Och99].

- `iterativo`. Corresponde al algoritmo *iterativo* propuesto en esta tesis y descrito en la sección 3.4. Realiza una distribución inicial de las palabras en C clases e iterativamente mueve palabras de unas clases a otras buscando un mejor agrupamiento.

- `iterativo_lo`. Corresponde al algoritmo *iterativo dejando uno fuera* propuesto en esta tesis y descrito en la sección 3.5. El modo de operar es idéntico al método anterior salvo por la optimización que se realiza utilizando la técnica de *dejar uno fuera*.

- `incremental_lo`. Corresponde al algoritmo *incremental* propuesto en esta tesis y descrito en la sección 3.6. Como método de optimización se utiliza el descrito en la sección 3.5 basado en la técnica de dejar uno fuera.

B.2.1. Experimentos con EuTrans I

La siguiente tabla muestra la perplejidad medida sobre el test para distintos modelos de lenguaje de la parte castellana del corpus EuTrans I. Los modelos de lenguaje comparados son modelos de bigramas basados en los agrupamientos proporcionados por distintos métodos de agrupamiento monolingüe. La perplejidad utilizando un modelo de bigramas sin utilizar clases toma el valor 8,59. (Ver figura 6.1.)

Clases	srilm_inc	srilm_full	mkcls	iter	iter_lo	inc_lo
100	9,08	8,81	8,78	9,14	9,10	9,07
150	8,22	7,98	8,00	8,11	8,06	8,09
200	7,72	7,71	7,77	7,83	7,75	7,73
250	7,72	7,75	7,78	7,94	8,05	7,69
300	8,09	8,12	8,06	8,20	8,14	7,82
350	8,36	8,77	8,54	8,47	8,21	7,86
400	8,57	8,52	8,38	8,44	8,25	7,95
450	8,50	8,49	8,52	8,34	8,38	8,02
500	8,51	8,50	8,71	8,58	8,51	8,12
550	8,54	8,54	8,55	8,60	8,44	8,25
600	8,56	8,56	8,55	8,58	8,37	8,28
650	8,58	8,58	8,57	8,57	8,53	8,47
686	8,60	8,60	8,59	8,60	8,60	8,60

La siguiente tabla muestra la perplejidad medida sobre el conjunto de entrenamiento de la parte castellana del corpus EuTrans I. La perplejidad sobre el entrenamiento utilizando un modelo de bigramas toma el valor 4,76. (Ver figura 6.2.)

Clases	iter	iter_lo	inc_lo
100	5,85	5,87	5,84
150	5,24	5,27	5,27
200	4,99	5,04	5,03
250	4,90	4,98	4,97
300	4,86	4,97	4,96
350	4,84	4,97	4,96
400	4,82	4,99	4,97
450	4,80	5,01	4,98
500	4,79	5,05	5,00
550	4,78	5,08	5,04
600	4,77	5,12	5,08
650	4,76	5,20	5,14
686	4,76	5,25	5,25

B.2.2. Experimentos con EuTrans II

La siguiente tabla muestra la perplejidad obtenida utilizando los modelos de bigramas basados en los agrupamientos proporcionados por los distintos métodos de agrupamiento descritos en la sección 6.2 utilizando el corpus EuTrans II (versión 5.1) sobre el idioma inglés. La perplejidad utilizando el modelo de bigramas es 15,06. (Ver figura 6.3.)

Clases	srilm_inc	srilm_full	mkcls	iter	iter_lo	inc_lo
200	15,94	15,90	15,58	15,84	16,03	15,81
400	15,25	15,36	15,23	15,50	15,35	15,41
600	15,13	15,18	15,15	15,46	15,43	15,27
800	15,15	15,12	15,13	15,30	15,42	15,24
1.000	15,10	15,10	15,10	15,22	15,25	15,18
1.200	15,10	15,09	15,11	15,42	15,40	15,20
1.400	15,12	15,13	15,09	15,12	15,10	15,13
1.600	15,08	15,08	15,07	15,06	15,04	15,06
1.700	15,06	15,06	15,05	15,04	15,06	15,04
1.701	15,06	15,06	15,06	15,06	15,06	15,06

La siguiente tabla muestra la perplejidad medida sobre el corpus de entrenamiento de los métodos propuestos en esta tesis. La perplejidad con el modelo de bigramas sobre el corpus de entrenamiento es 8,9. (Ver figura 6.4.)

Clases	iter	iter_lo	inc_lo
200	11,47	13,42	13,26
400	9,87	12,60	12,59
600	9,35	12,69	12,65
800	9,12	12,98	12,92
1.000	9,01	13,44	13,34
1.200	8,95	14,12	14,07
1.400	8,92	15,09	15,02
1.600	8,91	17,94	17,81
1.700	8,90	19,67	19,67
1.701	8,90	19,69	19,69

B.2.3. Agrupamiento bilingüe y colocaciones

B.2.4. Experimentos con EuTrans I

La siguiente tabla muestra los valores de BLEU y WER obtenidos tras entrenar OMEGA utilizando el modelo IBMm4 de alineamiento con distintas clases en el idioma origen y en el destino del corpus EuTrans I. Para cada número de clases del idioma origen se muestra el resultado correspondiente al número de clases en el idioma destino con el que se obtiene el mejor BLEU. (Ver figura 6.6.)

Orig	Dest	BLEU	WER	Orig	Dest	BLEU	WER
50	50	0,8780	7,05	400	200	0,8790	7,02
100	100	0,8774	7,06	450	200	0,8788	7,04
150	100	0,8772	7,11	500	200	0,8788	7,04
200	200	0,8791	7,00	550	200	0,8788	7,04
250	200	0,8788	7,02	600	200	0,8791	7,02
300	200	0,8785	7,04	650	200	0,8784	7,05
350	200	0,8785	7,04	686	200	0,8776	7,09

La siguiente tabla presenta los valores de BLEU y WER obtenidos utilizando OMEGA entrado con los agrupamientos generados por el método *incremental dejando uno fuera* con distintos valores del parámetro b. Los experimentos se han realizado variando el número de clases bilingües de 100 a 500 clases. Sólo se muestra los resultados para aquel número de clases con el que se obtiene el mejor BLEU (Ver figura 6.7.)

b	Clases	BLEU	WER	b	Clases	BLEU	WER
0,10	350	0,9385	3,49	0,55	375	0,9421	3,31
0,15	350	0,9383	3,54	0,60	375	0,9411	3,33
0,20	350	0,9388	3,49	0,65	375	0,9425	3,23
0,25	350	0,9365	3,62	0,70	375	0,9437	3,16
0,30	350	0,9384	3,51	0,75	375	0,9434	3,17
0,35	325	0,9401	3,34	0,80	400	0,9435	3,19
0,40	350	0,9401	3,37	0,85	400	0,9433	3,25
0,45	350	0,9400	3,39	0,90	375	0,9457	3,08
0,50	350	0,9426	3,23	0,95	400	0,9433	3,23

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo* para distintos números de clases utilizando el modelo IBMm2 de alineamiento. (Ver figura 6.8.)

Clases	BLEU	WER	SER	PER
–	0,8614	6,94	47,43	6,71
100	0,8961	5,77	38,12	5,32
125	0,8973	5,63	35,65	5,17
150	0,9142	4,94	30,57	4,66
175	0,9054	5,23	31,68	4,97
200	0,9073	5,28	31,58	4,96
225	0,9104	5,13	30,97	4,83
250	0,9060	5,24	32,78	4,96
275	0,9002	5,43	35,15	5,14
300	0,8943	5,82	36,88	5,50
325	0,8938	5,79	37,12	5,49
350	0,8899	5,77	38,65	5,54
375	0,8887	5,82	38,79	5,53
400	0,8872	5,95	38,99	5,68
425	0,8854	5,97	39,99	5,72
450	0,8841	6,04	40,05	5,79
475	0,8829	6,12	41,22	5,88
500	0,8747	6,34	43,09	6,07

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo* para distintos números de clases utilizando el modelo de alineamiento IBMm2 con vecinos. (Ver figura 6.8.)

Clases	BLEU	WER	SER	PER
–	0,8723	6,55	43,89	6,35
100	0,9017	5,47	37,38	5,16
125	0,8991	5,49	34,98	5,11
150	0,9208	4,36	28,50	4,10
175	0,9134	4,84	30,34	4,56
200	0,9160	4,75	28,47	4,51
225	0,9222	4,22	27,40	3,98
250	0,9184	4,38	29,31	4,15
275	0,9146	4,61	30,54	4,37
300	0,9099	4,77	32,18	4,56
325	0,9034	5,09	34,25	4,86
350	0,9036	5,12	34,58	4,87
375	0,8986	5,38	35,85	5,15
400	0,8965	5,47	36,48	5,25
425	0,8960	5,43	36,88	5,20
450	0,8936	5,55	37,35	5,32
475	0,8939	5,57	37,22	5,41
500	0,8899	5,75	39,19	5,58

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo* para distintos números de clases utilizando el modelo de alineamiento IBMm3. (Ver figura 6.8.)

Clases	BLEU	WER	SER	PER
–	0,8655	7,70	47,33	7,43
100	0,8987	5,56	38,12	5,20
125	0,8910	5,95	36,88	5,54
150	0,9164	5,05	30,47	4,65
175	0,9190	4,82	28,17	4,45
200	0,9206	4,70	27,80	4,41
225	0,9221	4,61	26,87	4,34
250	0,9242	4,48	26,34	4,25
275	0,9153	5,02	30,01	4,78
300	0,9107	5,25	31,88	5,00
325	0,9063	5,63	34,25	5,39
350	0,9072	5,41	34,25	5,15
375	0,9026	5,74	36,11	5,53
400	0,8971	6,00	37,82	5,77

continúa en la página siguiente ➪

viene de la página anterior

Clases	BLEU	WER	SER	PER
425	0,8963	6,01	37,85	5,78
450	0,8884	6,30	40,25	6,05
475	0,8864	6,45	40,59	6,23
500	0,8804	6,82	42,82	6,56

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo* para distintos números de clases utilizando el modelo de alineamiento IBMm4. (Ver figura 6.8.)

Clases	BLEU	WER	SER	PER
–	0,8791	7,00	43,36	6,78
100	0,8917	6,28	38,85	6,00
125	0,9106	5,00	33,31	4,77
150	0,9316	4,16	25,90	3,97
175	0,9320	4,01	25,73	3,84
200	0,9360	3,73	23,77	3,57
225	0,9338	3,87	23,63	3,70
250	0,9227	4,63	29,57	4,43
275	0,9174	4,89	31,58	4,68
300	0,9154	5,06	33,31	4,93
325	0,9135	4,96	34,21	4,84
350	0,9163	4,83	33,78	4,70
375	0,9099	5,43	35,15	5,27
400	0,9087	5,39	35,71	5,23
425	0,9074	5,59	36,35	5,42
450	0,9053	5,62	36,95	5,46
475	0,8919	6,47	40,52	6,29
500	0,8893	6,59	41,22	6,40

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm2. (Ver figura 6.9.)

Clases	BLEU	WER	SER	PER
–	0,8614	6,94	47,43	6,71
100	0,8904	6,39	38,68	5,82

continúa en la página siguiente

➪ viene de la página anterior

Clases	BLEU	WER	SER	PER
125	0,8916	5,86	38,35	5,56
150	0,9251	4,24	26,90	3,96
175	0,9218	4,47	27,90	4,17
200	0,9218	4,35	27,77	4,03
225	0,9233	4,34	28,17	4,08
250	0,9173	4,61	30,64	4,36
275	0,9175	4,49	30,31	4,21
300	0,9073	5,08	33,51	4,84
325	0,9062	5,12	34,38	4,84
350	0,9083	4,94	34,55	4,61
375	0,9051	5,24	35,25	4,98
400	0,9002	5,54	36,72	5,24
425	0,8940	5,80	39,02	5,43
450	0,8936	5,72	38,65	5,38
475	0,8914	5,81	39,55	5,49
500	0,8935	5,61	38,89	5,38

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm2 con vecinos. (Ver figura 6.9.)

Clases	BLEU	WER	SER	PER
–	0,8723	6,55	43,89	6,35
100	0,9039	5,28	36,08	4,94
125	0,9253	4,08	27,40	3,83
150	0,9284	4,03	25,90	3,60
175	0,9334	3,73	24,70	3,44
200	0,9336	3,60	24,17	3,38
225	0,9335	3,53	25,10	3,38
250	0,9298	3,70	26,57	3,54
275	0,9252	4,05	28,74	3,88
300	0,9244	4,07	28,91	3,89
325	0,9195	4,31	31,88	4,13
350	0,9198	4,20	31,24	4,04
375	0,9166	4,36	32,41	4,19
400	0,9086	4,90	34,65	4,68
425	0,9045	5,11	35,95	4,91

continúa en la página siguiente ➪

viene de la página anterior

Clases	BLEU	WER	SER	PER
450	0,9023	5,15	36,28	4,93
475	0,9045	5,16	35,45	5,02
500	0,9040	5,09	35,58	4,94

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm3. (Ver figura 6.9.)

Clases	BLEU	WER	SER	PER
–	0,8655	7,70	47,33	7,43
100	0,8698	7,20	45,63	6,76
125	0,9066	5,14	32,41	4,71
150	0,9112	5,02	31,07	4,63
175	0,9335	3,98	24,13	3,71
200	0,9343	3,88	23,53	3,57
225	0,9384	3,60	22,56	3,35
250	0,9302	4,17	25,60	3,85
275	0,9250	4,31	27,94	4,02
300	0,9223	4,53	28,91	4,24
325	0,9172	4,78	30,41	4,52
350	0,9174	4,68	30,91	4,39
375	0,9145	4,92	32,64	4,63
400	0,9095	5,12	34,18	4,84
425	0,9063	5,23	35,18	4,95
450	0,9051	5,50	35,38	5,25
475	0,9019	5,59	36,65	5,36
500	0,8994	5,82	37,28	5,63

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *iterativo dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm4. (Ver figura 6.9.)

Clases	BLEU	WER	SER	PER
–	0,8791	7,00	43,36	6,78
100	0,8942	5,86	37,32	5,57
125	0,9146	4,83	31,91	4,64

continúa en la página siguiente

➪ viene de la página anterior

Clases	BLEU	WER	SER	PER
150	0,9257	4,23	27,20	4,02
175	0,9376	3,63	22,86	3,48
200	0,9373	3,70	23,50	3,55
225	0,9344	3,91	24,63	3,74
250	0,9329	3,80	25,07	3,66
275	0,9274	4,06	27,27	3,94
300	0,9288	3,99	27,17	3,87
325	0,9227	4,53	30,84	4,37
350	0,9226	4,49	31,54	4,32
375	0,9185	4,69	32,94	4,52
400	0,9141	4,86	33,88	4,69
425	0,9104	4,93	35,25	4,77
450	0,9066	5,20	36,65	5,08
475	0,9090	5,12	36,28	5,01
500	0,9019	5,65	37,68	5,53

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *incremental dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm2. (Ver figura 6.10.)

Clases	BLEU	WER	SER	PER
–	0,8614	6,94	47,43	6,71
100	0,8932	6,03	38,42	5,53
125	0,8916	6,04	38,22	5,71
150	0,8973	5,70	35,98	5,43
175	0,9079	5,00	32,71	4,72
200	0,9189	4,46	28,64	4,21
225	0,9162	4,63	28,97	4,39
250	0,9231	4,17	27,40	3,91
275	0,9293	4,00	26,30	3,71
300	0,9272	4,05	27,10	3,80
325	0,9233	4,28	28,14	3,98
350	0,9233	4,26	28,14	3,96
375	0,9220	4,28	28,50	4,00
400	0,9221	4,33	27,77	4,05
425	0,9211	4,42	28,24	4,14
450	0,9179	4,50	29,14	4,21

continúa en la página siguiente ➪

viene de la página anterior

Clases	BLEU	WER	SER	PER
475	0,9189	4,43	28,37	4,22
500	0,9170	4,61	29,21	4,40

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *incremental dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm2 con vecinos. (Ver figura 6.10.)

Clases	BLEU	WER	SER	PER
–	0,8723	6,55	43,89	6,35
100	0,9151	4,67	32,38	4,30
125	0,9130	4,63	32,38	4,20
150	0,9215	4,32	30,57	4,03
175	0,9315	3,87	25,27	3,69
200	0,9345	3,65	23,63	3,42
225	0,9356	3,52	22,73	3,27
250	0,9346	3,68	24,50	3,48
275	0,9353	3,55	22,93	3,24
300	0,9319	3,71	24,97	3,50
325	0,9298	3,80	25,37	3,49
350	0,9297	3,79	25,40	3,56
375	0,9279	3,89	25,73	3,70
400	0,9297	3,77	25,03	3,60
425	0,9306	3,68	24,57	3,52
450	0,9266	3,81	25,97	3,63
475	0,9257	3,91	26,54	3,73
500	0,9237	4,09	27,30	3,88

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *incremental dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm3. (Ver figura 6.10.)

Clases	BLEU	WER	SER	PER
–	0,8655	7,70	47,33	7,43
100	0,8660	7,18	47,33	6,77
125	0,9097	4,89	33,85	4,55
150	0,9136	4,69	32,31	4,36

continúa en la página siguiente

➪ viene de la página anterior

Clases	BLEU	WER	SER	PER
175	0,9181	4,66	30,31	4,32
200	0,9309	4,01	25,43	3,66
225	0,9380	3,70	22,46	3,39
250	0,9415	3,39	21,36	3,17
275	0,9400	3,60	22,23	3,41
300	0,9341	3,90	24,00	3,67
325	0,9260	4,28	26,13	4,04
350	0,9255	4,32	26,34	4,09
375	0,9279	4,16	25,47	3,95
400	0,9247	4,42	26,84	4,21
425	0,9245	4,50	27,27	4,28
450	0,9223	4,50	27,80	4,28
475	0,9205	4,65	28,70	4,41
500	0,9176	4,93	29,74	4,69

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA sin clases y OMEGA con las clases generadas por el algoritmo *incremental dejando uno fuera* para distintos números de clases utilizando el modelo de alineamiento IBMm4. (Ver figura 6.10.)

Clases	BLEU	WER	SER	PER
–	0,8791	7,00	43,36	6,78
100	0,9043	5,11	35,05	4,89
125	0,9118	4,87	32,54	4,66
150	0,9175	4,64	29,74	4,48
175	0,9183	4,56	29,51	4,40
200	0,9303	3,94	25,20	3,80
225	0,9294	4,16	26,17	4,01
250	0,9373	3,83	24,40	3,68
275	0,9360	3,86	24,90	3,72
300	0,9409	3,34	21,70	3,21
325	0,9438	3,17	20,66	3,03
350	0,9450	3,13	20,39	2,99
375	0,9457	3,08	19,76	2,96
400	0,9437	3,24	20,69	3,12
425	0,9432	3,25	20,99	3,11
450	0,9363	3,63	23,40	3,50
475	0,9341	3,72	24,20	3,59

continúa en la página siguiente ➪

✣ viene de la página anterior

Clases	BLEU	WER	SER	PER
500	0,9285	3,99	26,07	3,85

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA para 1.000 a 10.000 frases de entrenamiento. (Ver figura 6.11.)

Frases	BLEU	WER	SER	PER
1.000	0,6806	18,95	82,18	18,17
2.000	0,7519	14,37	73,36	13,80
3.000	0,7863	12,38	67,19	12,00
4.000	0,8083	10,97	62,78	10,62
5.000	0,8283	9,58	57,01	9,29
6.000	0,8495	8,56	51,97	8,31
7.000	0,8595	7,88	48,50	7,65
8.000	0,8668	7,58	46,43	7,37
9.000	0,8733	7,33	45,09	7,13
10.000	0,8791	7,00	43,36	6,78

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *iterativo*. Para cada número de muestras, sólo se presentan los resultados obtenidos para el número de clases con el que se obtiene el mejor BLEU. (Ver figura 6.11.)

Frases	Clases	BLEU	WER	SER	PER
1.000	100	0,7381	15,98	74,07	15,29
2.000	150	0,8111	11,26	58,91	10,79
3.000	100	0,8498	9,17	52,04	8,67
4.000	175	0,8800	7,18	41,29	6,88
5.000	175	0,8978	6,04	35,21	5,84
6.000	175	0,9126	4,96	30,67	4,77
7.000	175	0,9206	4,57	28,24	4,38
8.000	175	0,9214	4,48	27,57	4,23
9.000	175	0,9325	3,78	24,23	3,66
10.000	200	0,9360	3,73	23,77	3,57

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *iterativo dejando uno fuera*. Para cada número de muestras, sólo se presentan los resultados obtenidos para el número de clases con el que se obtiene el mejor BLEU. (Ver figura 6.11.)

Frases	Clases	BLEU	WER	SER	PER
1.000	225	0,7426	15,45	71,23	14,95
2.000	200	0,8352	9,71	53,94	9,33
3.000	150	0,8689	7,94	45,83	7,44
4.000	200	0,8893	6,38	37,72	6,14
5.000	175	0,9061	5,53	33,24	5,29
6.000	200	0,9195	4,73	28,77	4,53
7.000	125	0,9243	4,46	27,87	4,25
8.000	175	0,9319	3,87	24,90	3,70
9.000	175	0,9371	3,60	23,10	3,50
10.000	175	0,9376	3,63	22,86	3,48

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera*. Para cada número de muestras, sólo se presentan los resultados obtenidos para el número de clases con el que se obtiene el mejor BLEU. (Ver figura 6.11.)

Frases	Clases	BLEU	WER	SER	PER
1.000	175	0,7747	13,84	67,76	13,27
2.000	225	0,8435	9,46	50,07	9,00
3.000	225	0,8801	7,15	40,72	6,81
4.000	225	0,9060	5,54	32,88	5,28
5.000	250	0,9177	5,04	30,94	4,85
6.000	250	0,9221	4,53	29,11	4,36
7.000	325	0,9271	4,21	27,24	4,05
8.000	300	0,9342	3,62	24,23	3,50
9.000	200	0,9397	3,37	21,53	3,28
10.000	375	0,9457	3,08	19,76	2,96

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera*. El número de clases óptimo se ha determinado de forma automática por el propio algoritmo de agrupamiento. (Ver figura 6.12.)

Frases	Clases	BLEU	WER	SER	PER
1.000	172	0,7751	13,76	67,72	13,21
2.000	213	0,8454	9,40	49,40	8,96
3.000	233	0,8809	7,10	40,22	6,74

continúa en la página siguiente ➪

viene de la página anterior

Frases	Clases	BLEU	WER	SER	PER
4.000	244	0,9008	5,88	35,68	5,68
5.000	260	0,9185	4,96	31,04	4,76
6.000	260	0,9263	4,43	27,27	4,23
7.000	259	0,9220	4,56	29,64	4,35
8.000	267	0,9314	4,06	26,17	3,88
9.000	263	0,9362	3,80	24,90	3,70
10.000	269	0,9360	3,83	25,03	3,69

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases y colocaciones para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera*. Para cada número de muestras, sólo se presentan los resultados obtenidos para el número de clases con el que se obtiene el mejor BLEU. (Ver figura 6.13.)

Frases	Clases	BLEU	WER	SER	PER
1.000	100	0,7889	13,12	63,99	12,17
2.000	200	0,8727	7,79	42,96	7,35
3.000	225	0,8961	6,63	38,08	6,29
4.000	225	0,9258	4,47	26,77	4,24
5.000	225	0,9350	3,83	24,03	3,64
6.000	250	0,9379	3,76	23,13	3,48
7.000	250	0,9458	3,09	19,53	2,95
8.000	250	0,9488	3,03	19,06	2,86
9.000	250	0,9538	2,58	17,36	2,45
10.000	250	0,9515	2,77	18,02	2,65

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA con las clases y colocaciones para distintos números de clases con 10.000 frases de entrenamiento. Como método de agrupamiento bilingüe se ha utilizado el algoritmo *incremental dejando uno fuera*. (Ver figura 6.14.)

Clases	BLEU	WER	SER	PER
100	0,9334	3,93	24,67	3,62
125	0,9362	3,83	23,93	3,53
150	0,9390	3,61	23,66	3,36
175	0,9369	3,94	23,16	3,63
200	0,9377	3,56	22,46	3,25

continúa en la página siguiente

➪ viene de la página anterior

Clases	BLEU	WER	SER	PER
225	0,9471	2,91	20,03	2,71
250	0,9515	2,77	18,02	2,65
275	0,9457	3,18	19,79	3,01
300	0,9472	2,98	19,43	2,83
325	0,9481	2,97	19,23	2,81
350	0,9473	3,01	19,19	2,85
375	0,9490	2,94	19,13	2,79
400	0,9508	2,92	18,93	2,82
425	0,9501	2,90	19,23	2,79
450	0,9453	3,15	20,93	3,06
475	0,9418	3,36	22,33	3,26
500	0,9395	3,51	22,83	3,41

La siguiente tabla muestra los valores de BLEU, WER, SER y PER obtenidos con OMEGA con clases y colocaciones para 1.000 a 10.000 frases de entrenamiento. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera*. El número de clases óptimo se ha determinado de forma automática por el propio algoritmo de agrupamiento. (Ver figura 6.15.)

Frases	Clases	BLEU	WER	SER	PER
1.000	171	0,7698	14,30	66,49	13,55
2.000	217	0,8695	7,80	43,93	7,42
3.000	227	0,8976	6,51	37,78	6,18
4.000	243	0,9256	4,48	26,74	4,25
5.000	246	0,9331	3,95	24,40	3,75
6.000	259	0,9401	3,48	22,46	3,26
7.000	262	0,9439	3,23	20,36	3,07
8.000	271	0,9479	3,09	19,63	2,96
9.000	280	0,9499	2,93	18,62	2,78
10.000	279	0,9469	3,10	19,69	2,94

B.2.5. Experimentos con EuTrans II

La siguiente tabla muestra los valores de BLEU y WER obtenidos tras entrenar OMEGA utilizando el modelo IBMm4 de alineamiento con distintas clases en el idioma origen y en el destino del corpus EuTrans II. Para cada número de clases del idioma origen se muestra el resultado correspondiente al número de clases en el idioma destino con el que se obtiene el mejor BLEU. (Ver figura 6.18.)

Orig	Dest	BLEU	WER
50	50	0,4851	35,84
100	600	0,4855	35,79
200	600	0,4812	36,01
600	50	0,4782	36,09
1.000	50	0,4767	36,37
1.400	50	0,4777	36,50

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,5. (Ver figura 6.19.)

Clases	BLEU	WER	SER	PER
–	0,4855	35,79	95,67	29,32
200	0,3972	42,37	98,33	34,09
400	0,4132	41,17	97,67	33,01
600	0,4232	40,60	98,00	33,25
800	0,4343	39,90	97,67	32,64
1.000	0,4430	38,80	97,67	31,80
1.200	0,4347	39,46	97,67	32,18
1.400	0,4431	38,70	95,33	31,38
1.600	0,4642	36,88	96,00	30,07
1.800	0,4621	37,04	96,00	30,37
2.000	0,4657	37,25	95,33	30,51
2.200	0,4659	36,93	95,33	30,35
2.400	0,4632	37,21	95,67	30,57
2.600	0,4662	37,39	96,00	30,57
2.800	0,4723	36,55	96,00	29,52
2.869	0,4855	35,79	95,67	29,32

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,7. (Ver figura 6.19.)

Clases	BLEU	WER	SER	PER
–	0,4855	35,79	95,67	29,32
200	0,4282	39,51	95,67	31,74
400	0,4215	40,30	98,33	31,99

continúa en la página siguiente ➪

➪ *viene de la página anterior*

Clases	BLEU	WER	SER	PER
600	0,4154	41,21	98,00	33,00
800	0,4348	39,33	97,00	31,96
1.000	0,4286	39,91	97,00	32,57
1.200	0,4258	39,64	98,00	32,61
1.400	0,4410	38,13	95,67	31,01
1.600	0,4430	38,09	96,33	31,34
1.800	0,4461	38,53	95,33	31,96
2.000	0,4500	38,26	96,00	31,23
2.200	0,4659	36,85	95,67	30,28
2.400	0,4753	36,45	95,67	29,99
2.600	0,4718	37,03	96,00	30,19
2.800	0,4748	36,56	95,67	29,90
2.869	0,4855	35,79	95,67	29,32

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,9. (Ver figura 6.19.)

Clases	BLEU	WER	SER	PER
–	0,4855	35,79	95,67	29,32
200	0,4120	41,13	96,00	32,64
400	0,4265	39,82	96,67	32,27
600	0,4165	40,80	97,67	32,58
800	0,4268	39,78	98,00	31,78
1.000	0,4207	40,01	97,67	32,24
1.200	0,4368	39,38	97,00	31,89
1.400	0,4408	38,60	96,00	31,80
1.600	0,4368	39,51	95,00	32,14
1.800	0,4639	37,21	95,33	30,32
2.000	0,4687	37,03	95,67	30,21
2.200	0,4661	37,79	95,00	30,57
2.400	0,4705	37,08	95,67	30,00
2.600	0,4737	36,71	96,00	30,03
2.800	0,4759	36,48	95,67	29,86
2.869	0,4855	35,79	95,67	29,32

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases y colocaciones sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,5. (Ver figura 6.20.)

Clases	BLEU	WER	SER	PER
–	0,4694	37,17	95,67	30,62
200	0,4041	42,04	96,00	32,86
400	0,4276	40,12	96,33	32,14
600	0,4342	39,44	96,00	32,04
800	0,4209	39,67	96,33	32,11
1.000	0,4124	40,67	96,00	32,85
1.200	0,4361	39,35	96,67	32,20
1.400	0,4173	40,09	97,00	32,67
1.600	0,4362	39,04	97,00	31,85
1.800	0,4162	40,62	95,67	32,94
2.000	0,4408	39,00	96,33	31,80
2.200	0,4521	38,19	96,00	31,46
2.400	0,4580	37,64	95,67	31,05
2.600	0,4633	37,80	95,00	31,16
2.800	0,4601	37,77	95,00	31,27
3.000	0,4684	37,14	95,00	30,88
3.200	0,4636	37,25	95,33	30,88
3.400	0,4787	36,57	95,33	30,04
3.511	0,4694	37,17	95,67	30,62

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases y colocaciones sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,7. (Ver figura 6.20.)

Clases	BLEU	WER	SER	PER
–	0,4694	37,17	95,67	30,62
200	0,4085	41,64	96,00	33,43
400	0,4007	41,93	95,67	33,48
600	0,4031	41,32	96,00	33,09
800	0,4096	41,57	95,67	32,94
1.000	0,4155	40,91	95,33	32,80
1.200	0,4035	41,53	95,67	33,33
1.400	0,4076	41,45	96,33	33,95
1.600	0,4232	40,48	96,67	32,78
1.800	0,4133	40,08	95,67	32,72
2.000	0,4407	38,49	96,00	31,59
2.200	0,4492	38,89	95,67	31,78
2.400	0,4577	38,37	95,00	31,46

continúa en la página siguiente ➪

➪ *viene de la página anterior*

Clases	BLEU	WER	SER	PER
2.600	0,4632	37,73	95,00	31,33
2.800	0,4639	37,11	95,00	31,02
3.000	0,4661	37,21	95,00	31,06
3.200	0,4700	36,89	95,00	30,24
3.400	0,4756	36,75	95,33	30,04
3.511	0,4694	37,17	95,67	30,62

La siguiente tabla muestra el BLEU, WER, SER y PER obtenidos por OMEGA y por OMEGA con clases y colocaciones sobre el corpus EuTrans II. Como método de agrupamiento se ha utilizado el algoritmo *incremental dejando uno fuera* con b igual a 0,9. (Ver figura 6.20.)

Clases	BLEU	WER	SER	PER
–	0,4694	37,17	95,67	30,62
200	0,4012	43,06	96,33	33,76
400	0,4087	41,49	95,67	33,07
600	0,4026	41,72	96,33	33,62
800	0,4152	40,19	97,00	32,20
1.000	0,4274	39,36	96,67	31,66
1.200	0,4182	39,50	96,33	31,73
1.400	0,4229	39,54	95,33	31,46
1.600	0,4387	38,89	95,33	31,53
1.800	0,4392	38,22	96,00	31,26
2.000	0,4415	38,30	96,00	31,52
2.200	0,4381	38,30	94,67	31,49
2.400	0,4664	36,96	95,33	30,33
2.600	0,4653	37,50	94,67	31,11
2.800	0,4644	37,71	95,33	31,12
3.000	0,4615	37,50	95,33	31,01
3.200	0,4693	36,86	95,33	30,28
3.400	0,4745	36,92	95,33	30,39
3.511	0,4694	37,17	95,67	30,62