Enhancing Scene Text Recognition with Visual Context Information

By

Ahmed Sabir

Computer Science Department
Universitat Politècnica de Catalunya

Advisors
Professor Lluís Padró
Doctor Francesc Moreno-Noguer

A thesis submitted for the degree of

Doctor of Philosophy

Barcelona, September 2020

This thesis is submitted to the Computer Science Department, Universitat Politècnica de Catalunya, in fulfilment of the requirements for the degree in Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Ahmed Sabir, September 2020

Acknowledgements

I am exceptionally thankful for the guidance and support I have had from my two advisors Lluís Padró and Francesc Moreno-Noguer. They have both given me a great deal of independence in pursuing my ideas and contributed towards their development. I would like to thank also my previous advisors in Japan, Professor Michiko Matsuda, Yasuhiro Sudo from Mustda AI Lab and particularly Professor Hiroshi Tanaka who introduced me to academic research. Also, to Professor Jaime Lopez for his wise advice, and fruitful comments and discussions every Friday at university cafeteria. It also would never have been possible to complete this work without the amazing community feedback, reviewers and conference mentors. I thank my fellow office-mates at UPC-Ω Vahid and Christian for the stimulating discussions, for the sleepless nights we were working together before deadlines and for all the fun we have had. I would like to thank my friends, Rosa García and her family, Marina, Mateo, and Brian, for making Barcelona feel like my second home. Last, words fail to convey the gratitude that I feed towards my parents, who have done the impossible to make this possible.

This work described in this thesis was partially supported by KASP Scholarship Program and by the MINECO project HuMoUR TIN2017-90086-R. Any opinions, findings, and conclusion or recommendations expressed below are those of the author and not don't necessarily reflect the the view of KASP Foundation.

Abstract

This thesis addresses the problem of improving text spotting systems, which aim to detect and recognize text in unrestricted images (*e.g.* a street sign, an advertisement, a bus destination, etc.). The goal is to improve the performance of off-the-shelf vision systems by exploiting the semantic information derived from the image itself. The rationale is that knowing the content of the image or the visual context can help to decide which words are the correct candidate words. For example, the fact that an image shows a coffee shop makes it more likely that a word on a signboard reads as *Dunkin* and not *unkind*.

We address this problem by drawing on successful developments in natural language processing and machine learning, in particular, learning to re-rank and neural networks, to present post-process frameworks that improve state-of-the-art text spotting systems without the need for costly data-driven re-training or tuning procedures.

Discovering the degree of semantic relatedness of candidate words and their image context is a task related to assessing the semantic similarity between words or text fragments. However, semantic relatedness is more general than similarity (*e.g.* *car*, *road*, and *traffic light* are related but not similar) and requires certain adaptations. To meet the requirements of these broader perspectives of semantic similarity, we develop two approaches to learn the semantic relatedness of the spotted word and its environmental context: word-to-word (object) or word-to-sentence (caption). In the word-to-word approach, word embedding based re-rankers are developed. The re-ranker takes the words from the text spotting baseline and re-ranks them based on the visual context from the object classifier. For the second, an end-to-end neural approach is designed to drive image description (caption) at the sentence-level as well as the word-level (objects) and re-rank them based not only on the visual context but also on the co-occurrence between them.

As an additional contribution, to meet the requirements of data-driven approaches such as neural networks, we propose a visual context dataset for this task, in which the publicly available COCO-text dataset [Veit et al. 2016] has been extended with information about the scene (including the objects and places appearing in the image) to enable researchers to include the semantic relations between texts and scene in their Text Spotting systems, and to offer a common evaluation baseline for such approaches.

Resum

Aquesta tesi aborda el problema de millorar els sistemes de reconeixement de text, que permeten detectar i reconèixer text en imatges no restringides (per exemple, un cartell al carrer, un anunci, una destinació d'autobús, etc.). L'objectiu és millorar el rendiment dels sistemes de visió existents explotant la informació semàntica derivada de la pròpia imatge. La idea principal és que conèixer el contingut de la imatge o el context visual en el que un text apareix, pot ajudar a decidir quines són les paraules correctes. Per exemple, el fet que una imatge mostri una cafeteria fa que sigui més probable que una paraula en un rètol es llegeixi com a *Dunkin* que no pas com *unkind*. Abordem aquest problema recorrent a avenços en el processament del llenguatge natural i l'aprenentatge automàtic, en particular, aprenent re-rankers i xarxes neuronals, per presentar solucions de postprocés que milloren els sistemes de l'estat de l'art de reconeixement de text, sense necessitat de costosos procediments de reentrenament o afinació que requereixin grans quantitats de dades.

Descobrir el grau de relació semàntica entre les paraules candidates i el seu context d'imatge és una tasca relacionada amb l'avaluació de la semblança semàntica entre paraules o fragments de text. Tanmateix, determinar l'existència d'una relació semàntica és una tasca més general que evaluar la semblança (per exemple, *cotxe*, *carretera* i *semàfor* estan relacionats però no són similars) i per tant els metodes existents requereixen certes adaptacions. Per satisfer els requisits d'aquestes perspectives més àmplies de relació semàntica, desenvolupem dos enfocaments per aprendre la relació semàntica de la paraula reconeguda i el seu context: paraula-a-paraula (amb els objectes a la imatge) o paraula-a-frase (subtítol de la imatge). En l'enfocament de paraula-a-paraula s'usen re-rankers basats en word-embeddings. El re-ranker pren les paraules proposades pel sistema base i les torna a reordenar en funció del context visual proporcionat pel classificador d'objectes. Per al segon cas, s'ha dissenyat un enfocament neuronal d'extrem a extrem per explotar la descripció de la imatge (subtítol) tant a nivell de frase com a nivell de paraula i re-ordenar les paraules candidates basant-se tant en el context visual com en les co-ocurrencies amb el subtítol.

Com a contribució addicional, per satisfer els requisits dels enfocs basats en dades com ara les xarxes neuronals, presentem un conjunt de dades de contextos visuals per a aquesta tasca, en el què el conjunt de dades COCO-text disponible públicament [Veit et al. 2016] s'ha ampliat amb informació sobre l'escena (inclosos els objectes i els llocs que apareixen a la imatge) per permetre als investigadors incloure les relacions semàntiques entre textos i escena als seus sistemes de reconeixement de text, i oferir una base d'avaluació comuna per a aquests enfocaments.

Contents

List of Figures

Chapter 1

Introduction

Reading letters and words is an essential task in today's society. Written and printed texts are everywhere, in form of newspapers, documents, text in the wild, etc. The machine replication of human vision, like reading, has been a dream of scientists and researchers for a long time. Over the last five decades, machine-reading and computer vision application has grown from a dream to a reality. However, research in some areas, such as *text recognition* in the wild, has not reached a mature enough level of implementation, and there are still many challenges due to the many possible variations in textures, backgrounds, fonts, and lighting conditions that are present in such images. Scanned document recognition has grown rapidly on different applications, and paper digitalization and camera based applications are everywhere. The success of the Optical Character Recognition system (OCR) has a long history in computer vision. However, the success of the OCR system is restricted to scanned documents. Scene texts exhibit a large variability in appearances, and can prove to be challenging even for the state-of-the-art OCR methods. Many scene understanding methods recognize objects and regions like roads, trees, or the natural scene (*i.e.* sky) in the image successfully, but tend to neglect the text on the signboards. In this work our goal is to fill this gap in understanding the scene. In addition, the automatic detection and recognition of text in natural images, *text spotting*, is an important step towards visual understanding.

There are several areas where text spotting and recognition system aids are needed, such as screen readers that can help blind users and those with low vision to access documents [Nazma et al. 2016]. There are a small number of systems [Merler et al. 2007, Rajkumar et al. 2014] that can provide reliable access to the printed text on common household items such as product packages and prescription medication bottles. Figure 1.1 shows several examples of texts in the wild.

Figure 1.1: Text is found in a variety of different formats and scenarios in images. The variety of textures, fonts, orientations, noise, objects, and background are some of the challenges associated with capturing scene text in the wild. However, this is a useful problem to solve, as often it is the text contained that captures the semantics of an image.

Another area where complex background text spotting and recognition might be of great use is in autonomous driving [Wang et al. 2012]. For example, an autonomous car driving system may need to read signs to understand the rules of the road to assist the driver [Priambada and Widyantoro 2017].

A further useful application of text detection and recognition is image retrieval. In this case, the system would be able to search for specific text in large scale image searches. Also, text spotting can be used to retrieve specific text in an image in a big database; for example, the system can search for a text in a video or TV recording library and retrieve specific material [Jaderberg et al. 2014c].

1.1 Motivation

Existing approaches to scene text recognition (a.k.a Text Spotting) usually divide the problem into two fundamental tasks: 1) *text detection*, consisting of selecting the image regions likely to contain texts, and 2) *text recognition*, converting the images within these bounding boxes into a readable string.

Text spotting[1] may be tackled either by lexicon-based or lexicon-free approaches. Lexicon-based recognition methods use a pre-defined dictionary as a reference to guide the recognition. Lexicon-free methods (or unconstrained recognition techniques), predict character sequences without relying on any dictionary.

Most recently, deep learning approaches are dominating the state-of-the-art method in text recognition. Nevertheless, current state-of-the-art deep learning

[1]Here we mean text recognition.

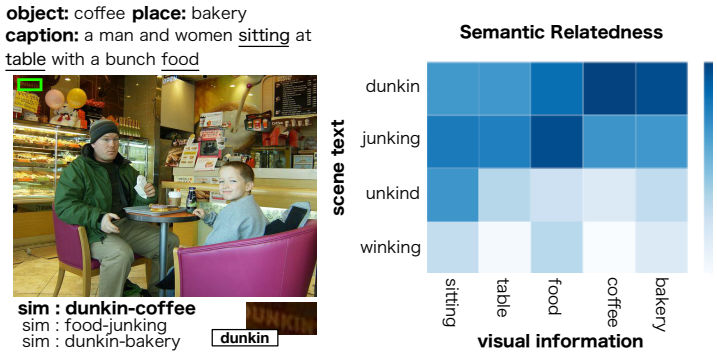

Figure 1.2: Overview of the natural language understanding visual context information. The word "dunkin" has a strong semantic relation with "bakery", "food", and "coffee", thus it will be more likely to appear in this image than other similar words, such as "junking" or "unkind".

methods, whether lexicon-based or lexicon-free, have some drawbacks: Lexicon-based approaches need a large dictionary to perform the final recognition. Thus, their accuracy will depend on the quality and coverage of this lexicon, which makes this approach unpractical for real world applications where the domain may be different to the one the system was trained on. On the other hand, lexicon-free recognition methods rely on language models to predict character sequences, and they may therefore generate likely sequences that do not correspond to actual words in the language. In both cases, these techniques rely on the availability of large datasets for training and validation, and these may not be always available for the target domain.

The interest of the computer vision community in Text Spotting has significantly increased in recent years. However, state-of-the-art scene text recognition methods [Liao et al. 2019, Xing et al. 2019] do not leverage object and scene semantic information. Scene text, objects appearing in the image, and the overall scene are closely connected with each other, particularly with the semantic relativity between the spotted text and its image context. For example, as shown in Figure 1.2 the word "dunkin" has a stronger semantic relation with "coffee" and "bakery", and it will therefore be more likely to appear in this visual context than other possible candidates such as "junking" or "unkind". This thesis addresses one of the main limitations of scene text recognition, *i.e.* lack of semantic understanding.

1.2 Objective

The goal of this thesis is to establish a new framework for improving text recognition in scene text images by incorporating the semantic relationships present among the text, background objects and scene data. This framework is applicable to any text recognition architecture that outputs multiple word/text hypotheses with associated scores or probabilities. These individual word scores are modified (re-ranked) based on the contextual information in the image, such as objects present, scene/location, or the caption of the picture. Each kind of context information is extracted through the appropriate state-of-the-art frameworks.

1.3 Contributions

The main contributions of the thesis may be summarized as follows:

A. **Word-to-Word Re-ranker.** We present post-processing approaches to improve scene text recognition accuracy by using the occurrence probabilities of words (unigram language model), and the semantic correlation between scene and text.

B. **Word-to-Sentence Re-ranker.** We propose a simple, end-to-end deep learning architecture to learn the semantic relatedness of word-to-word, word-to-sentence pairs and show how it outperforms other semantic similarity scorers when used to re-rank candidate answers in the Text Spotting task.

C. **Textual Visual Semantic Dataset.** We present a visual context dataset for the text spotting problem. Unlike other methods that use complex architecture to extract visual information, our approach utilizes out-of-the-box state-of-the-art tools. Therefore, the dataset annotation can be improved in the future when better systems become available. This dataset can be allowed to leverage semantic relations between image context and candidate texts into text spotting systems, either as post-processing or in end-to-end training approaches. This is the first visual context dataset for text spotting and also the first dataset that approaches this problem with simple data extraction.

1.4 Structure of the Thesis

The thesis is organized according to the following chapters:

- **Chapter 2** introduces the foundation of text spotting problem and semantic similarity. In particular, we describe several recent works in both stand-alone text spotting and visual context-based model. We highlight the most recent state-of-the-art model in semantic similarity in a word-level with word embedding and sentence-level with deep learning.

- **Chapter 3** describes the proposed dataset used in this thesis. First, we report an overview of the publicly available dataset for this problem. Then, we introduce our methods to extract visual context from pre-existing dataset. Precisely, we introduce multiple visual classifiers to extract the most related visual to the image. Finally, we also present a small survey and a report for human evaluation on this dataset.

- **Chapter 4** introduces the proposed word-embedding based re-ranker framework. We describe the proposed visual context pipeline integration into the text spotting system. Two similarity-to-probability conversion methods [Blok et al. 2003] are proposed and analyzed from the belief-revision viewpoint, since our re-rankers can be seen as a revision of text hypothesis probabilities based on the visual context observed in the image.

- **Chapter 5** focuses on the proposed neural based re-ranker framework. The techniques proposed in previous Chapter 4 are based on word-level framework, allowing to integrate any visual context in a word-level manner. In this chapter, we exploit, sentence-level, additional visual information obtained from textual descriptions of the image (caption). We also introduce a new deep learning architecture to perform the fusion of several re-rankers.

- **Chapter 6** summarizes this thesis challenges and achievements, and outlines possible future research directions.

1.5 Publications

This section gives a list of publications that contain the core of this thesis. The thesis is divided in three parts. The first part contains the introduction of the main

core concept of applying natural language processing tools to text spotting system, as well as the word-to-word re-ranker model in Chapter 4. This part is based on the following two publications:

- Sabir et al. (2018a) **Enhancing Text Spotting with a Language Model and Visual Context Information**. In *International Conference of the Catalan Association for Artificial Intelligence (CCIA)*, 2018. (National Conference).
- Sabir et al. (2018b) **Visual Re-ranking with Natural Language Understanding for Text Spotting**. In *Asian Conference on Computer Vision (ACCV)*, 2018.

The second part extended the framework into sentence-level with end-to-end word-to-sentence re-ranker as in Chapter 5.

- Sabir et al. (2019) **Semantic Relatedness Based Re-ranker for Text Spotting**. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
- Sabir et al. (forthcoming) **Context-aware Text Recognition in the Wild**. Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

Finally, the third part deals with the construction of the dataset that used in this thesis, which is described in Chapter 3.

- Sabir et al. (2020) **Textual Visual Semantic Dataset for Text Spotting**. In *The Conference on Computer Vision and Pattern Recognition Workshop on Text and Documents in the Deep Learning Era (CVPR-W)*, 2020.

Chapter 2

Literature Review

In this chapter we review several background research related to this thesis. We start with a review of Text Spotting in Section 2.1, as well as a number of works combining language and vision in scene text recognition in Section 2.2. We review the development of learning semantic similarity in Section 2.3. In Section 2.4 we provide an overview of most recent text modelling approaches for semantic similarity. In Section 2.5 we review a range of different methods for text modeling with deep learning. In Section 2.6 we provide an overview of the most recent approaches in semantic similarity with contextual word embeddings.

2.1 Text Spotting

The *text spotting* task consists of recognizing pieces of text appearing in images in the wild (*e.g.* ads or traffic signs in a street or commercial brands on product labels in a supermarket). This task is still an ongoing problem in computer vision and state-of-the-art results still fall short of the high performance achieved by well-established OCR systems. This problem can be tackled from either a lexicon-based or a lexicon-free approach. Lexicon-based recognition methods use a pre-defined dictionary as a reference to guide recognition. Lexicon free methods (or unconstrained recognition techniques), predict characters without relying on a dictionary. The first lexicon-free text spotting system was proposed by [Neumann and Matas 2010]. The system extracted character candidates via maximally stable extremal regions (MSER) and eliminated non-textual ones through a trained classifier. The remaining candidates were fed into a character recognition module, trained using a large amount of synthetic data.

More recently, several deep learning alternatives have been proposed as shown in Table 2.1. For instance, PhotoOCR [Bissacco et al. 2013] uses a Deep Neural

Table 2.1: Quantitative comparison of existing text spotting algorithms.

Algorithm	Positive	Negative
MSER [Neumann and Matas 2010]	First approach end-to-end system	Only horizontal texts
CNN [Wang et al. 2012]	Robust/Good performance	Only horizontal texts
CRF [Mishra et al. 2012]	Uses both top-down/bottom-up cues	Complex background
PhotoOCR [Bissacco et al. 2013]	Robust/fast/good performance	Huge training dataset
Strokelets [Yao et al. 2014]	Good on different orientations text	Complex background
CNN [Jaderberg et al. 2016]	Robust/90k dictionary	Fixed lexicon
CNN-RNN-CTC [Shi et al. 2016]	Capture long sequence	CTC loss complexity
CNN-LSTM [Ghosh et al. 2017]	Visual attention	Short words
CNN-CTC [Gao et al. 2017]	Handel both fixed/free based lexicon	CTC loss complexity
CNN-LM [Fang et al. 2018]	Excellent on complex natural images	Rely on language model
CNN-Chart [Xing et al. 2019]	End-to-end in one stage	Small bounding box

Network (DNN) that performs end-to-end text spotting using histograms of oriented gradients as input. It is a lexicon-free system that is able to read characters in uncontrolled conditions. The final word selection is performed by means of two language models, namely a character and a word N-gram language model, where the character 8-gram model, re-ranked with the probability, learns from the second-level word language model. Another approach also employed a language modelling for final word selection [Mishra et al. 2012]. Top-down integration can tolerate errors in text detection and mis-recognition.

The first attempt to use a Convolutional Neural Network (CNN) was proposed by [Wang et al. 2012] and was pre-trained with unsupervised learning features. The word ranking score is based on post-processing techniques, such as Non-Maximal Suppression (NMS) and beam search. Another CNN based approach is that of [Jaderberg et al. 2016], which applies a sliding window over CNN features that use a fixed-lexicon. This is further extended in [Jaderberg et al. 2014c], through a deep architecture that allows feature sharing. Another method by the same authors [Jaderberg et al. 2014a] proposes two different CNNs that rely on character-level models. The first CNN uses a character sequence model, and the second model uses a bag-of-n-grams encoding model. In [Shi et al. 2016] the problem is addressed using a Recurrent CNN, a novel lexicon-free neural network architecture that integrates Convolutional and Recurrent Neural Networks for image based sequence recognition. Ghosh et al. (2017) use visual attention mechanism [Xu et al. 2015] to guide the recognition through an LSTM [Hochreiter and Schmidhuber 1997] decoder. In particular, they use the CNN-90k model [Jaderberg et al. 2016] as an encoder but without the final layer to extract the most import feature vectors from each text image. Gao et al. (2017) introduce a CNN with Connectionist Temporal Classification (CTC) [Graves et al. 2006] to generate the final label sequence

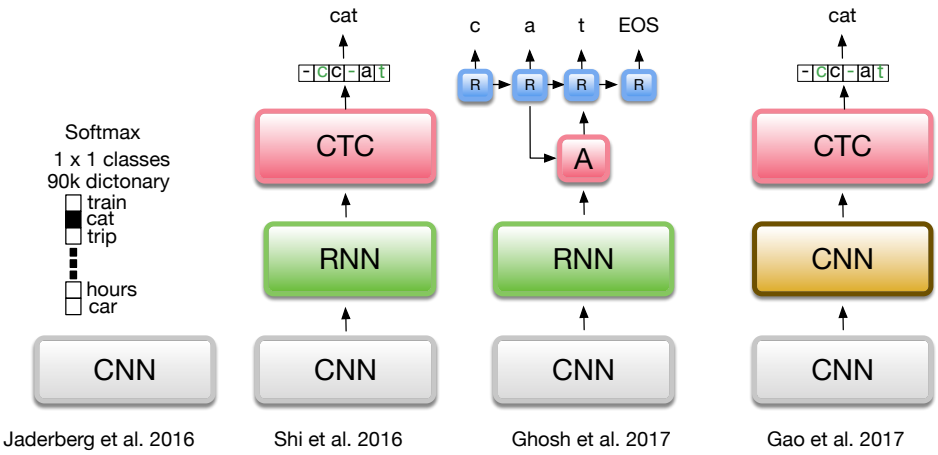

Figure 2.1: Four typical deep learning-based architectures for text spotting. The figure shows the last stage of text spotting where the texts are predicted.

without a sequence model such as RNN or LSTM. This approach uses stacked convolutional to capture the dependencies of the input sequence. This algorithm can be integrated into both fixed lexicon and lexicon free based recognition models. More recently, Fang et al. (2018) introduced a CNN based encoder-decoder architecture. The decoder uses a language model and an attention mechanism, in an ensemble manner, for the character sequence generation. Xing et al. (2019) present Convolutional Character Networks (CharNet) as end-to-end[1] (detection and recognition in one stage). The CharNet directly produces the word characters from the bounding box. Unlike other methods, this approach is trained on both synthetic and real-world data.

Figure 2.1 summarizes the most typical deep learning-based approaches for text spotting. All models are trained on the same synthetic word dataset [Jaderberg et al. 2014b].

Most recent state-of-the-art systems adopt deep learning based approaches [Shi et al. 2016, Jaderberg et al. 2016, Ghosh et al. 2017, Gao et al. 2017, Fang et al. 2018]. However, despite very promising results, these approaches have some limitations: the accuracy of lexicon-based approaches depends on the quality and coverage of the used lexicon, which makes them unpractical for many real applications. In addition, lexicon-free methods rely on neural language models to predict character sequences, and may therefore generate probable character combinations that do not correspond to actual words. Both approaches rely on the availability of large training datasets, which may not be always available for the target domain. Also,

[1] In this thesis, we are interested in the last stage, where the text is predicted

some recent work [Shi et al. 2016, Gao et al. 2017] uses CTC loss [Graves et al. 2006] to compute the conditional probability of the target sequence label, which can be very expensive to compute and is more difficult to optimize than a general loss function (*i.e.* CTC spiky distribution problem) [Miao et al. 2015].

2.2 Visual Context for Text Spotting

Objects appearing in the image and the overall scene are often related to each other. Therefore, not only understanding the visual environment around the text is very important for recognizing an object in images, but also this semantic relations could be exploited in the opposite direction (*i.e.* employ object information to improve scene text recognition). This has been explored recently by a relatively reduced number of works. For example, Zhu et al. (2016) show that the visual context could be beneficial for text detection. This work uses a 14-class pixel classifier to extract *context features* from the image, such as *tree, river, wall*, to then assist scene text detection. For example, if the classifier output is *tree, river*, the probability of no-text is very high. While if the background is a *signboard*, the classifier will try to locate and detect the text. Although this method uses understanding at the pixel-level without any semantic relations, it shows that combining computer vision and natural language processing methods can offer promising results.

Kang et al. (2017) employ topic modeling to learn the correlation between visual context and the spotted text in social media. The metadata associated with each image (*e.g.* tags, comments and titles) are then used as context to enhance recognition accuracy. In other words, metadata help to narrow down the candidate words that are more likely appear in those images. Another work that uses prior visual information to improve performance in the text spotting task is [Patel et al. 2016]. Patel *et al.* use Latent Dirichlet Allocation (LDA) [Blei et al. 2003] to generate a new lexicon as shown in Figure 2.2. The topic modeling learns the relation between text and images. This approach, however, relies on captions describing the images rather than using the main key words semantically related to the images to generate the lexicon re-ranking. Thus, the lexicon generation can be inaccurate in some cases due to the unrelated image captions to the images (false description), especially in cases with complex background or out of context scenarios images (*e.g.* car in the river, TV in the street).

Karaoglu et al. (2017b) take advantage of text and the visual context to tackle the logo retrieval problem. This approach, however, uses textual cues to retrieve

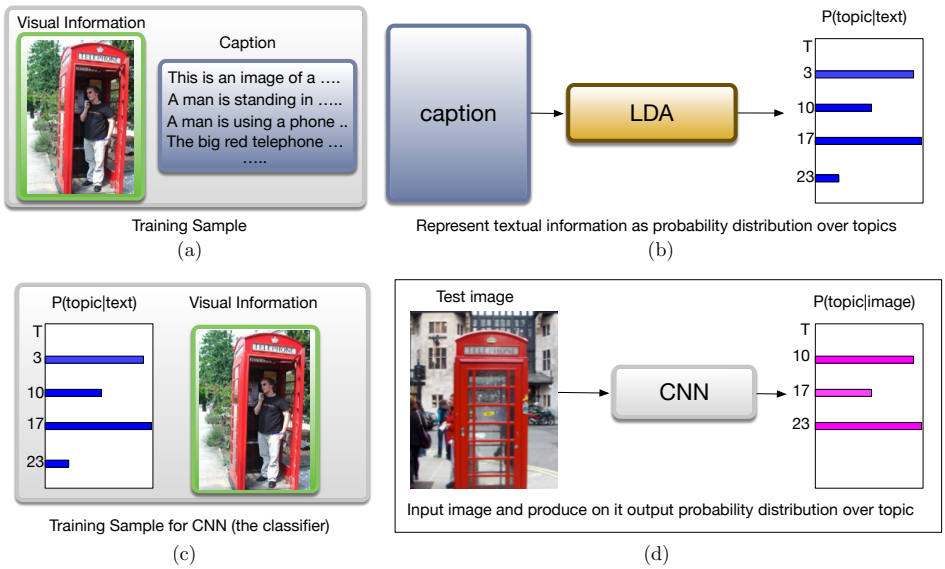

Figure 2.2: Overview of dynamic text generation [Patel et al. 2016]. (a) Training sample from the COCO-caption dataset. (b) The use of top modeling to represent the textual information as a probability distribution $P(topic|text)$. (c) Training sample after presenting each caption with topics (stage b). (d) The trained classifier (CNN) takes an image as input and produces its distribution over topics $P(topic|text)$. Figure adopted from Patel et al. (2016).

logos based on the relation between the text that appears in the logo and the logo itself. For example, finding the character trigrams *sta, tar* raises the probability of there being *starbucks* logo would be higher than any other, as shown in Figure 2.3. The same authors [Karaoglu et al. 2017a] also use word-level approach based on the CNN-90k dictionary [Jaderberg et al. 2016] for the fine-grained classification and logo retrieval. Bai et al. (2018) proposed an approach that combines visual and text features in images for fine-grained classification. This work uses the semantic relationships between textual and visual cues to enhance image classification and image retrieval. They use an attention mechanism to find out relevant words and eliminate irrelevant ones by using visual context information. However, the limitation of this approach is that it depends on one visual cue, which may result in false-positive results in complex background scenarios.

Most recently, Prasad and Wai Kin Kong (2018) use object information surrounding the spotted text to guide text detection. They introduce a dataset containing over 22k natural images (*i.e.* non-synthetic) with around 277k bounding boxes for text and 42 text-related object classes. They propose two sub-networks to learn the relation between text and object class (*e.g.* relations such as *car–plate* or *sign_board–digit*). However, the semantic relation is limited to 42 predefined object

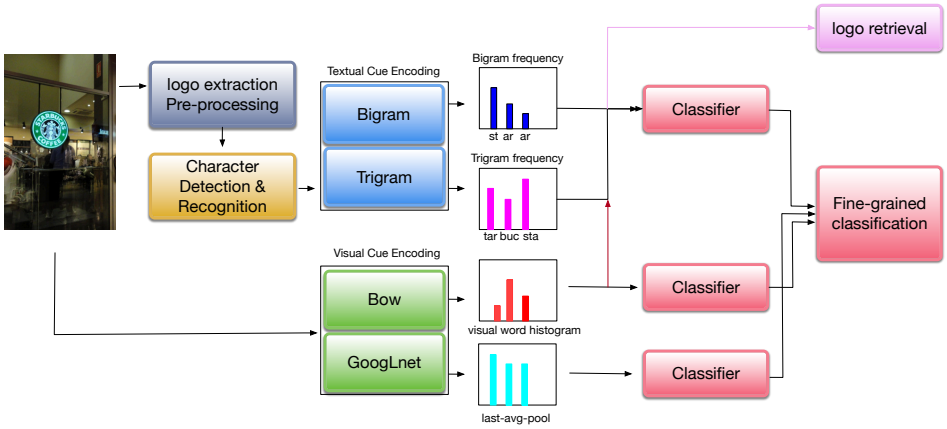

Figure 2.3: Overview of the method proposed by [Karaoglu et al. 2017b]. They combine textual (bigrams, trigrams) and visual cues for fine-grained classification and logo retrieval. Figure adopted from Karaoglu et al. (2017b).

classes. Although there is no semantic or natural language understanding between the object class and the detected text, the proposed method and their results show that the visual context (*e.g.* object information) gives a boost to the performance of text spotting system.

Unlike these methods, our approaches in Chapter 4 and Chapter 5 use direct visual context from the image where the text appears, and do not rely on any extra resource such as human labeled meta-data [Kang et al. 2017] nor do they limit the context object classes [Prasad and Wai Kin Kong 2018]. In addition, our approach is easy to train and can be used as a drop-in complement for any text-spotting algorithm that outputs a ranking of word hypotheses. In addition, our method uses multiple visual context information unlike [Bai et al. 2018] and [Patel et al. 2016] that rely on only one source of visual context, which may result in false positives in cases with complex backgrounds.

2.3 Semantic Similarity

Semantic similarity measure aims to capture how close are two text fragments (*i.e.* words, phrases, sentences) via a numerical score based on how close their respective meanings are. Semantic relatedness is a concept more general than similarity, assuming that two concepts may be related but not necessarily similar (*e.g.* *car* and *traffic light* are not *similar*, but they are clearly *related*). In computational linguistics, semantic relatedness is defined for any two concepts that have any kind of semantic relation [Budanitsky and Hirst 2001]. Semantic similarity is more specific

and is a particular case of semantic relatedness [Turney and Pantel 2010]. Semantic similarity methods can be divided into two categories: corpus-based methods and knowledge-based methods [Mihalcea et al. 2006, Zhu and Iglesias 2016]. Corpus-based methods are a common approach where word relations are learned from a general corpus, while knowledge-based approaches require building and using extra resources like WordNet [Miller 1998], BabelNet [Navigli and Ponzetto 2012], FrameNet [Baker et al. 1998], etc.

2.3.1 Knowledge-based Methods

Knowledge-based semantic similarity methods use ontologies or other knowledge repositories (*e.g.* Wiki) to measure semantic similarity between words. The similarity measure between two words is determined by how close they are in the graph defined by the reference ontology or repository. WordNet [Miller 1998] is often used as a reference graph, since it is organized in *synsets* (sets of synonym words) which are connected via different semantic relations (hypernymy, meronymy, anotnymy, etc) to form a graph. Since each word may have different meanings, and thus, belong to more than one synset, knowledge-based similarity methods need to either disambiguate the right sense for the words (which is not always easy or possible) or check for the interpretation of each of the words that yields maximum similarity score [Sánchez et al. 2012]. Let $s(w)$ denote the set of synsets to which word w belongs, then the word similarity sim between two words is defined as:

$$\mathrm{sim}_{word}(w_1, w_2) = \max_{a \in s(w_1), b \in s(w_2)} \mathrm{sim}_{synset}(a, b) \tag{2.1}$$

where $\mathrm{sim}_{synset}(a, b)$ is the synset similarity, often based on the shortest-path distance [Budanitsky and Hirst 2006] between synsets a and b in the semantic graph used as reference. There is a variety of semantic similarity measures traditionally used on WordNet – although many of them could be applied to other graphs or semantic resources.

In particular, Knowledge-based similarity methods exploit the taxonomic or hierarchical structure provided by class/subclass (or hypernym/hyponym) relationships. The basic intuition is that two concepts close in a taxonomy (*e.g.* *lion* and *tiger*) are also semantically close. So, a basic similarity metric could be:

$$\mathrm{sim}_{spl}(a, b) = SPL(a, b) \tag{2.2}$$

where $SPL(a, b)$ is the *shortest path lenght* from a to b.

However, this approach ignores some aspects of taxonomical relationships such as heterogeneous granularity (*e.g.* the semantic distance of *living thing* and *animal* is very large compared to that between *cat* and *kitten*, even if graph distance is 1 in both cases). To take this into account, other measures can be used:

$$\text{sim}_{lch}(a,b) = -\log \frac{SPL(a,b)}{2MaxDepth} \tag{2.3}$$

where $MaxDepth$ is the maximum depth of the taxonomy. Another approach in a similar fashion [Wu and Palmer 1994] tries to measure the depth of the two concepts with respect to their last common sub-summer node (LCS):

$$\text{sim}_{wup}(a,b) = 2\frac{\text{depth}(LCS(a,b))}{\text{depth}(a)+\text{depth}(b)} \tag{2.4}$$

where $\text{depth}(a)$ is the distance of node a to the taxonomy root.

Recently, most works in knowledge-based methods are based on a neural network that embed words into low-dimensional vector spaces, from text corpora –*i.e.* word embeddings. Iacobacci et al. (2015) introduce a knowledge-based word2vec, that uses an external lexical database such as BabelNet [Navigli and Ponzetto 2012]. These models are trained on large corpus annotated with synsets that link each word to a concept in the the database. We discuss sense embedding more in detail in Section 2.4.7.

2.3.2 Corpus-based Methods

Corpus-based semantic similarity methods are based on word associations learned from large text collections. The two words are considered similar based on their most frequent surrounding contexts. The relation is learned based on word distribution in the corpus or word co-occurrences. Two possible metrics to compute semantic similarity are:

Pointwise Mutual Information (PMI) [Turney 2001]: PMI is an unsupervised measure for word semantic similarity evaluation. The model is based on word co-occurrence count from large corpora. For two words w_1 and w_2, PMI is measured as:

$$PMI\left(w_1,w_2\right) = \log_2 \frac{p\left(w_1 \& w_2\right)}{p\left(w_1\right)p\left(w_2\right)} \tag{2.5}$$

where $p\left(w_1 \& w_2\right)$ is the probability that w_1 and w_2 co-occur in the same document.

Latent Semantic Analysis (LSA) [Landauer et al. 1998]: LSA is another method that uses corpus based measure for semantic similarity. The co-occurrences of

word count are captured by means of a dimensionality reduction by singular value decomposition (SVD) on a words×documents matrix representing the corpus. SVD is simple linear algebra operation, that can be applied to any rectangular matrix to find the correlations among its rows and columns. LSA is an approach that is able to overcome some limitation of the sparseness and high dimensionality of the standard vector space model. Therefore, the similarity is computed in the lower dimensional space, where the second order relations between the terms or concepts are exploited. Thus, the resulting vectors can be used to compute standard cosine similarities. The LSA similarity [Mihalcea et al. 2006] can be computed as the following scoring function:

$$\text{sim}\,(T_1, T_2) = \frac{1}{2}\left(\frac{\sum_{w\in T_1} idf(w)\cdot \max_{x\in T_2}\text{sim}(w,x)}{\sum_{w\in T_1} idf(w)} + \frac{\sum_{w\in T_2} idf(w)\cdot \max_{x\in T_1}\text{sim}(w,x)}{\sum_{w\in T_2} idf(w)}\right) \quad (2.6)$$

where T_1 and T_2 are the sentences or text fragments to compare. Given the two text fragments T1 and T2, this formula combines the semantic similarity of individual words in each text with words in the other text segment. Similarity values range between 0 and 1, being 0 the value for no semantic similarity or overlapping, and 1 indicating maximum similarity. Each text is represented in LSA space by summing up the normalized vectors of all words or as inverse document frequency *idf*, weighting scheme as show in Equation 2.6 giving more significance to a specific word than to a generic one –*i.e. bring*, *get*, etc.

Most recent works are based on advanced computational techniques, such as word2vec [Mikolov et al. 2013b] and GloVe [Pennington et al. 2014] that use neural networks to compute embeddings able to represent words with low dimensional vectors. In this thesis, we follow this approach as it achieves state-of-the-art results in many Semantic Textual Similarity tasks (STS). We describe in more detail the recent advancement in word embedding model in Section 2.4.

2.3.3 Similarity Measure

Once we have words represented as vectors in a semantic space, semantic similarities can be derived from vector distance measurements. In this section, we will describe these different distance measures that are able to measure the similarity distance between two dense vectors.

2.3.3.1 Minkowski Distance

Minkowski is a family of distance measures that includes two particular cases [Cha 2007]: Euclidean distance and Manhattan distance. The Minkowski distance is defined as follows:

$$d_{min} = \left(\sum_{i=1}^{n} |x_i - y_i|^m \right)^{\frac{1}{m}}, m \geq 1 \tag{2.7}$$

where m is a real number and x_i and y_i are the two vectors in dimension space. For $m = 1$ we obtain Manhattan distance, and $m = 2$ yields Enclidean distance.

2.3.3.2 Manhattan Distance

The Manhattan distance is a particular case of the Minkowski when m=1. The Manhattan distance is defined as:

$$d_{\text{man}} = \sum_{i=1}^{n} |x_i - y_i| \tag{2.8}$$

2.3.3.3 Euclidean Distance

The most commonly used distance for numerical data is the Euclidean distance. This is a particular case of the Minkowski when m=2. Although the Euclidean distance is used in many clustering algorithms [Xu and Wunsch 2005], it has a major drawback when comparing vectors that have a similar distance or have smaller attribute values. The Euclidean distance is defined as:

$$D_{ij} = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^{1/2} \right)^2 \tag{2.9}$$

2.3.3.4 Cosine Measure

The Cosine similarity measure is most commonly used for document similarity and word-sentence similarity. The cosine similarity is defined as:

$$\text{cosine}(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\|x\|_2 \|y\|_2} \tag{2.10}$$

where $\|y\|_2$ is the Euclidean norm of vector $y = (y_1, y_2, ..., yi)$ that defined as:

$$\|y\|_2 = \sqrt{y_1^2 + y_2^2 + \ldots + y_n^2} \tag{2.11}$$

2.3.4 Similarity to Probability

The similarity measure that we describe in the previous section is not suitable for word re-ranking. Therefore, in order to be able to re-rank words based on their probability, some strategy to convert similarity to probability is needed.

In this section, we discuss the theories behind similarity to probability conversion. Blok et al. (2003) introduce a conditional probability model that assumes the preliminary probability result is updated or revised to the degree that the hypothesis proof warrants. The range of revision is based on the informativeness of the argument and its degree of similarity. That is, the similarity to probability conversion can be defined in terms of **belief revision**. Belief revision is a process of forming a belief by bringing into account a new piece of information.

1. `Tigers` **`can bite through wire,`** `therefore`
   `Jaguars` **`can bite through wire.`**

2. `kittens` **`can bite through wire,`** `therefore`
   `Jaguars` **`can bite through wire.`**

Consider the observations (1) and (2). In the first case, the statement seems logical because it matches our expectations (jaguars are similar to tigers, so we expect them to be able to do similar things), so the statement is consistent with our previous belief, and there is no need to revise it. In the second case, the statement is surprising because our prior belief is that kittens are not so similar to jaguars, and thus, not so strong. But if we assume the veracity of the statement, then we need to revise and update our prior belief about kitten strength.

Blok et al. (2003) formalizes belief as probabilities and revised belief as conditional probabilities, and provides a framework to compute them based on the similarities of the involved objects. According to the authors, belief revision should be proportional to the similarity of the involved objects (*i.e.* in the example, the statement about kittens and jaguars would cause a stronger belief revision than *e.g.* the same statement involving a pigeons and jaguars because they are less similar).

In our case, we use the same rationale and the same formulas to convert similarity (or relatedness) scores into probabilities suitable for reranking.

2.3.4.1 SimProb Model

To convert the similarity to probability, or SimProb model [Blok et al. 2007], we need three parameters:

- **Hypothesis**: prior probabilities.
- **Informativeness**: conclusion events.
- **Similarities**: measure the relatedness between involved categories.

The main goal is to predict a conditional probability of statements, given one or more others. In order to predict the conditional probability of the argument's conclusion, given its premise or hypothesis, we will need only the prior probabilities of the statements, as well the similarities between the involved categories (*e.g.* kittens and tigers).

Single-Premise Formulation of SimProb

The conditional probability $P(Q_c|Q_a)$ is expressed in terms of the prior probability of the conclusion statement $P(Q_c)$, the prior probability of the premise statement $P(Q_a)$, and the similarity between the conclusion and the premise categories $\mathrm{sim}(a, c)$.

$$P(Q_c|Q_a) = P(Q_c)^{\alpha} \tag{2.12}$$

where:

- **Hypothesis**: $P(Q_c)$
- **Informativeness**: $1 - P(Q_a)$
- **Similarities**: $\alpha = \left[\frac{1 - \mathrm{sim}(a,c)}{1 + \mathrm{sim}(a,c)}\right]^{1-P(Q_a)}$

There are two factors to determine the hypothesis probability revision: 1) the sufficient relatedness to the category –as $sim(a, c)$ goes to 0, α goes to 1, and thus $P(Q_c|Qa) = P(Q_c)$, *i.e.* no revision takes place (no changes in the original belief), while as $sim(a, c)$ goes to 1, α goes to 0, and the hypothesis probability $P(Q_c)$ is revised and raised closer to 1. And 2) the informativeness of the new information $1 - P(Q_a)$ –as $P(Q_a)$ approaches 1 and in consequence is less informative, α also goes to 1, since there is no new information, and thus no revision is needed either.

Two-Premise Formulation of SimProb

When we have more than one evidence supporting the revision, we can reason using Equation 2.13 below. In this scenario we begin with the conditional probability that comes from the dominant premise alone $P(Q_c|Q_a)$ (let's assume Q_a is dominant). Then we add a fraction of the remaining lack of trust or confidence $1 - P(Q_c|Q_a)$ that the dominant conditional *leaves behind*. The similarity between premise categories defines the size of the portion or fraction $sim(a, b)$ and the separate the influence of the nondominant premise on the conclusion prior $P(Q_c|Q_b) - P(Q_c)$. The component of similarity is designed to reduce the impact of the non-dominant premise when the premises are redundant. Note that the proposed method in Equation 2.13 guarantees an increase in strength with additional premises. In addition, this property, $\Pr(Qc|Qa, Qb) \geq \Pr(Qc|Qa), \Pr(Qc|Qb)$, is noncompetitive in sense that one category does not reduce probability of concept by another.

So, the revision formula used in this case is:

$$P(Qc|Qa, Qb) = \beta M + (1 - \beta)S$$

where:

$$\begin{aligned} \beta &= \max \left\{ \begin{array}{l} \mathrm{sim}(a, b) \\ \mathrm{sim}(a, c) \\ \mathrm{sim}(b, c) \\ 1.0 - \mathrm{sim}(a, c) \\ 1.0 - \mathrm{sim}(b, c) \\ P(Q_a) \\ P(Q_b) \end{array} \right\} \\ M &= \max\{P(Q_c|Q_a), P(Q_c|Q_b)\} \\ S &= P(Q_c|Q_a) + P(Q_c|Q_b) - P(Q_c|Q_a) \times P(Q_c|Q_b) \end{aligned} \tag{2.13}$$

where $P(Q_c|Q_a)$ and $P(Q_c|Q_b)$ are defined by Equation 2.12.

2.4 Word Embeddings

We now describe the component of word-to-dense vector, in particular, word embedding model. As we explained in the previous sections, to make an algorithm understand any form of free text, we need to convert that text into numeric values. The most commonly used and simplest way is to do this is to convert the word into **one-hot-encoding** in which each distinct word stands for a binary number that becomes a vector with one dimension. However, one-hot-encoding to encode

an entire vocabulary is impractical and computationally expensive, as the dimensions of these representations are enormous. Word embedding solves this problem by representing a word in a non-binary vector with lower and denser dimensions.

2.4.1 Overview

The most general word embedding is trained in an unsupervised way without any specific object and instead aims to capture language knowledge from a large corpus [Mikolov et al. 2013a]. These word vectors are trained using co-occurrence information, which is called a pre-trained word vector. The pre-trained word vector can be utilized for vector feature extraction or to measure the distance similarity between two words as a stand-alone model.

Since word2vec was introduced, it has been the focus of much ongoing research in the community. Pennington et al. (2014) introduced a model that can derive better semantic relationships between words from both the co-occurrence matrix and the incorporation of global co-occurrence statistics. Kusner et al. (2015) proposed a Word Mover's Distance (WMD) model that takes advantage of the semantic relation that word2ve vectors often preserve during the training, (*i.e.* v(Barcelona)-v(London)+ v(Paris). The model utilizes word2vec to compute the distance between document$_1$ and document$_2$ so that it can measure even uncommon words as shown in Figure 2.4. Joulin et al. (2017) present an n-gram character-based word embedding that can deal with out-of-vocabulary (OOV) and rare words during training. Camacho et al. (2019) introduced relational word embeddings, and an enhanced version that encodes complementary relational knowledge to the standard word vector in the semantic space.

In the line of employing embedding based approaches or pre-trained word embedding vectors, many methods have proposed complex neural architectures that are able to exploit word vectors successfully. One such architecture is Recurrent Neural Networks, Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997], which is able to capture historical information from different parts of input sequences. The architecture can handle sequential information as the current input can access the previous output. This concept of adopting different sentence sizes is essential in natural text processing, as the text can be modelled as a sequence of characters or as words or sentences. Mueller and Thyagarajan (2016) tackle the semantic similarity task by the applying the Siamese Recurrent Network based on LSTM to estimate the degree of similarity between two sentences.

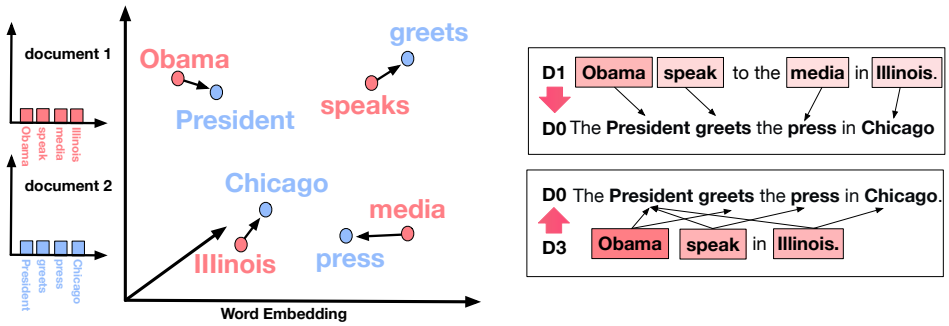

Figure 2.4: Illustration of an application that depends on word embedding to leverage semantic similarity, Word Mover's Distance model [Kusner et al. 2015]. The closeness of the word pairs from document$_1$ and document$_2$ are: sim(Obama, President); sim(speaks, greets), sim(media, press) and sim(Illinois, Chicago). Figure reproduced from Kusner et al. (2015).

Siamese LSTMs used the same weight to encode each sentence representation and then predicted how close the sentence pairs were by the Manhattan distance.

Another neural architecture that is able to take advantage of pre-trained word embedding vectors is Convolutional Neural Network (CNN) architecture. CNNs have achieved many successes in many problems in computer vision, such as handwriting recognition [LeCun et al. 1989], text recognition [Wang et al. 2012], and object recognition [He et al. 2016]. In particular, after the success of CNN for image classification, Kalchbrenner et al. (2014) introduced a 1D CNN for sentence modeling and Kim (2014) for sentiment classification. The main idea of 1D Convolutional is sliding a k-widows over a sequence to extract different information such as n-grams. Kim (2014) introduced a shallow architecture that achieves state-of-the-art performance in sentiment analysis. Kim-CNN proposes fine-tuning the pre-trained embedding during training and sliding multiple kernels to extract different information from the sequence, improving the modelling outcome.

In this thesis, we employ both word embedding approaches as 1) standalone in Chapter 4, and as 2) vector feature extraction (fine-tune embedding) in Chapter 5 to learn both semantic similarity and semantic relatedness.

2.4.2 Learning Word Embeddings

The main objective of building a word embedding, is that it can learn the semantic similarity between words expected to be close in low dimension vector space. It also learns the hidden semantic relation, as disclosed by contextual information,

Table 2.2: The Skip-Gram model [Mikolov et al. 2013b] is trained to predict the probability of a word being a context word for the given target. Example demonstrates multiple pairs of target and context words as training samples, generated by a [5-word window] sliding along the sentence, as shown in this example **a man is holding tennis racket.**

Sliding window	Context
a man is	man, is
a man is holding	a, is, holding
a man is holding tennis	a, man, holding, tennis
a man is holding tennis racket	a, man, is, tennis, racket
...	..

between a word that appears in similar contexts. Learning word embedding is carried out by two methods:

Count-based Method: A count-based method is an unsupervised approach based on a matrix factorization of word, a co-occurrence matrix and raw co-occurrence counts. In particular, it relies heavily on word frequency, and the co-occurrence matrix, with the assumption that words that occur in the same contexts share a similar or related semantic meaning. Probabilistic Language Models [Bengio et al. 2003], are examples of count-based methods that map co-occurrences knowledge between neighboring words into low dimensions dense word vector, thereby *solving the curse of dimensionality*.

Context-based Method: This method is a supervised approach, giving the model the local context to predict the target word. Word-embedding is based on this approach that learns efficient word representation.

In this thesis, we built our system upon the second approach: context-based, prediction-based word-embedding.

2.4.3 Word2vec: Skip-Gram Model

The Skip-Gram model [Mikolov et al. 2013b] is a context-based feed-forward neural network model trained to predict the probability of a word being a context word for the given target. As shown in Table 2.2 each context-target pair (target in blue color) is a new observation in the training data. For example, the target word *holding* produces five training examples: (holding, a), (holding, man), (holding, is), (holding, tennis) and (holding, racket). As shown in Figure 2.5 (Left), given the vocabulary size V, we learn the word embedding vectors of size N. The model learns to predict the output of *context word* using the input *target word* at a time.

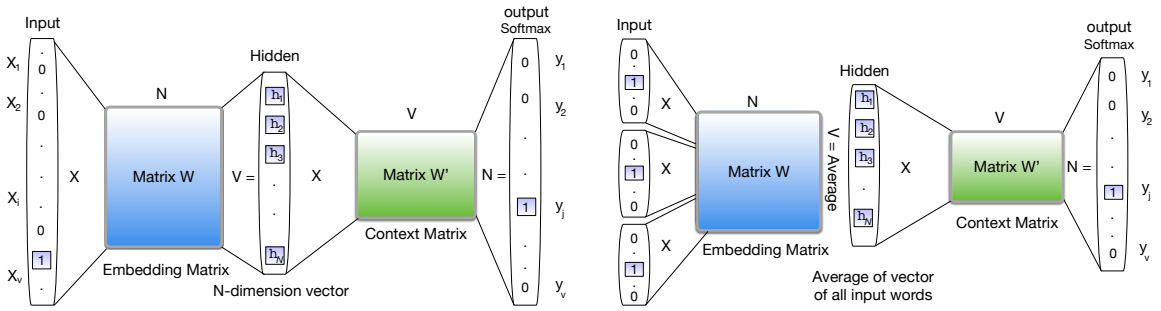

Figure 2.5: (Left) The Skip-Gram model. (Right) In the CBOW model, multiple context word vectors are averaged to produce the fixed-length vector in the hidden layers. The input and the output of both models are one-hot encoded word representations.

2.4.4 Word2vec: Continuous Bag-of-Words

The Continuous Bag-of-Words (CBOW) [Mikolov et al. 2013a] is similar to skip-gram, but predicts the target word based on the context words from the source sentence. The model takes the current middle word as input and tries to predict the context words correctly, as shown in (Right) Figure 2.5. As there are multiple contextual words, they average the corresponding word vector constructed by the multiplication of the matrix W and the input vector.

2.4.4.1 Loss Functions

The skip-gram model is trained to minimize a loss objective function. Several loss functions can be incorporated to train these language models, such as noise-contrastive estimation [Gutmann and Hyvärinen 2012] and negative sampling. In this section, we discuss the loss function *negative sampling* that were presented in the original paper [Mikolov et al. 2013b] and work well with a small dataset.

Negative Sampling

The skip-gram models as in Figure 2.5 (Left) define the embedding vector of each word by the matrix W and the context vector by the output matrix W'. Given an input word W_I as v_{w_I} the embedding vector and its corresponding context W' as v'_{w_I} the final output applies a softmax that predicts the output word w_O given w_I as in Equation 2.15. But first, let's define the Skip-gram loss function as:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p\left(w_{t+j}|w_t\right) \tag{2.14}$$

where T is the number of words, c is the context size window, w_t is the center word, and $p(w_{t+j}|w_t)$ is the final output layer that applies a softmax to compute the probability of predicting the output word w_O given w_I:

$$p\left(w_O|w_I\right) = \frac{\exp\left(v'^{\top}_{w_O} v_{w_I}\right)}{\sum_{i=1}^{V}\exp\left(v'_{w_i} T v_{w_I}\right)} \tag{2.15}$$

where v_w (input) and $v\prime_w$ (output) are the vector representation of w, and W matrix is the number of words in the vocabulary, as shown in the Skip-Gram model in Figure 2.5. Nonetheless, when V (vocabulary size) is very large, it is computationally inefficient for measuring the denominator by going through all of the terms for each single sample. The need for a more effective calculation of conditional probability leads to new approaches like Negative Sampling. The negative sampling [Mikolov et al. 2013b] maximizes the log probability of the softmax, and proposed replacing $\log p\left(w_O|w_I\right)$:

$$\log\sigma\left(v'^{\top}_{w_O} v_{w_I}\right) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma\left(-v^{\top}_{w_i} v_{w_I}\right)\right] \tag{2.16}$$

where k is the negative samples for each data sample. The task is to distinguish the target word w_O from a randomly picked word in the noise distribution $P_n(w)$ using a logistic regression. The noise distribution is defined as:

$$P_n(w) = U(w)^{3/4}/Z \tag{2.17}$$

where $U(w)$ is the uniform distributions and Z is the normalization factor that make the probability distribution of range [0,1]. The author claims that the ratio of $U(w)^{3/4}$ yielded the best result.

In summary, the basic concept of negative sampling is that for training, some false labels are needed to make the prediction. Therefore, they pick some non-surrounding words; calling it the k negative word as the false label sample. Note that, if the predicting word is picked from negative sampling, the model tries to choose another word again.

2.4.4.2 Subsampling of Frequent Words

Word frequency is too general to differentiate the context, since uncommon words are more likely to carry distinct or important information. To balance the frequent and the uncommon (rare) words, the author of word2vec [Mikolov et al. 2013b] proposed discarding words w with the probability of:

$$1 - \sqrt{t/f(w)} \tag{2.18}$$

during the sampling, where $f(w)$ is the word frequency and t is an adjustable threshold.

2.4.4.3 Finding Phrases in Text

Some phrases stand as conceptual units rather than individual simple words. Therefore, learning phrases such as *New York* at first and then learning the word embedding model improves the outcome quality. The basic concept is data-driven based on a unigram and bigram frequency count as:

$$s_{\text{phrase}} = \frac{C\left(w_i w_j\right) - \delta}{C\left(w_i\right) C\left(w_j\right)} \tag{2.19}$$

where C is the simple count of an unigram w_i or bigram w_j and δ is a parameter to prevent infrequent word and phrases. Higher scores indicate that higher probabilities of being phrases.

2.4.5 FastText

FastText [Joulin et al. 2017] is an extension of the word2vec model. The main concept is that instead of learning the word directly as word2vec, an n-gram character representation is learned. Thus, it can deal with rare words not seen during training by breaking them down to the character n-grams level to obtain their embeddings. The use of n-gram instead of bag-of-words has certain advantages, such as adding extra partial information about the local word order. Also, n-grams can explicitly embed morphology information, while, word2vce ignores the morphology of words. In other words, the model generates not only better embedding but also handles out-of-vocabulary (OOV) cases.

2.4.6 GloVe: Global Vectors

The GloVe Vector model [Pennington et al. 2014] aims to combine both context-based skip-gram model and the count-based matrix factorization approaches. As in the co-occurrences based model, the word count can reveal the meaning or senses of words. To distinguish from $p(w_o|w_I)$ in the word embedding that we described above, we defined the co-occurrence probability as:

$$p_{\text{co}}\left(w_k|w_i\right) = \frac{C\left(w_i, w_k\right)}{C\left(w_i\right)} \tag{2.20}$$

where $C(w_i, w_k)$ counts the co-ocurrence between w_i and w_k. For example, we have two words (w_i = bank) and (w_j= street) and the third word ($\tilde{w}_k$ = cash) is related to **bank**, and thus we expect $p_{\text{co}}(cash|bank)$ to be bigger than $p_{\text{co}}(cash|street)$.

The main concept here is that the word embeddings (*i.e.* skip-gram) are captured by the co-occurrence probabilities ratios rather than the probabilities themselves. The global vector shapes the relationship between two words regarding the third context word as:

$$F\left(w_i, w_j, \tilde{w}_k\right) = \frac{p_{\text{co}}\left(\tilde{w}_k|w_i\right)}{p_{\text{co}}\left(\tilde{w}_k|w_j\right)} \tag{2.21}$$

Additionally, since the aim is to learn meaningful word representation, F is intended to be a function of the linear difference between two words $w_i - w_j$:

$$F\left(\left(w_i - w_j\right)^\top \tilde{w}_k\right) = \frac{p_{\text{co}}\left(\tilde{w}_k|w_i\right)}{p_{\text{co}}\left(\tilde{w}_k|w_j\right)} \tag{2.22}$$

With the consideration of F being symmetrical between target words and context words, the final solution is to model F as an exponential function.

2.4.6.1 Loss Functions

The GloVe model loss function is designed to minimize the sum of the squared errors:

$$\mathcal{L} = \sum_{i=1,j=1}^{V} f\left(C\left(w_i, w_j\right)\right)\left(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log C\left(w_i, \tilde{w}_j\right)\right)^2 \tag{2.23}$$

where V is the size of vocabulary. The bias b_i is added to capture $-\log C$, where $\tilde{b}_j$ is added for $\tilde{w}_j$ to restore the symmetric forum. The function $f(c)$ should saturate when c becomes extremely large; and close to zero. The $f(c)$ weighting is adjustable function of the co-occurrence of w_i and w_j. The weighting function is defined as:

$$f(c) = \begin{cases} \left(\frac{c}{c_{\max}}\right)^{\alpha} & \text{if } c < c_{\max}, c_{\max} \text{ is adjustable} \\ 1 & \text{if otherwise} \end{cases} \tag{2.24}$$

2.4.7 Word2vec: Sense Embedding

One of the limitations of word embedding that has been mentioned above is that it computes a single representation for each word independently from the meaning they have in each context, thus collapsing all meanings of polysemous words in a single vector. In contrast, sense embeddings assign a vector to each word sense [Iacobacci et al. 2015, Pilehvar and Collier 2016, Camacho-Collados et al. 2016, Mancini et al. 2017, Iacobacci 2019]. These approaches rely on a sense-annotated corpus from sense inventory, such as WordNet [Miller 1998] and BabelNet[2] [Navigli and Ponzetto 2012].

2.5 Deep Learning for Semantic Similarity

The word embedding models described in the previous sections enable the comparison of word semantics. However, language is more usually used as text fragments (phrases, sentences) that need to be assigned similarities. This section describes learning at the sentence level via a multi-layer neural network. The general structure of a multi-layer neural network provides a powerful tool or framework for building many NLP applications, such as question answering [Severyn and Moschitti 2015], machine translation [Bahdanau et al. 2014] and image captioning [Vinyals et al. 2015]. This section focuses on a review of previous work and the basic block of deep learning for semantic similarity.

2.5.1 Convolutional Neural Network

The convolutional neural network is a multilayer, hierarchical neural network. Three principal factors distinguish the CNNs from the simple feed-forward neural networks described above 1) local receptive fields, 2) weight sharing, and 3) pooling or sub-sampling.

The deep structure of the CNNs allows them to refine feature representation and abstract semantic meaning gradually. CNNs have achieved many successes in many problems such as text recognition [Wang et al. 2012], object recognition

[2]http://www.babelnet.org/

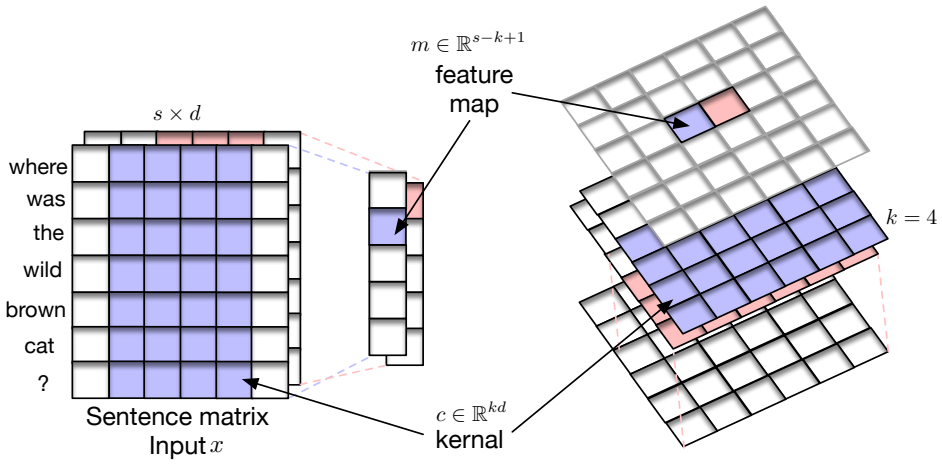

Figure 2.6: Illustration of 1-D convolutional layer mapping the sentence to their feature representations (feature map).

[He et al. 2016], and text classification [Conneau et al. 2016]. CNNs based method was the first work by [LeCun et al. 1998] to apply CNNs as a classifier, in a sliding window fashion, for digit recognition task. The structure of the convolution kernel, that simulate the locally-sensitive and orientation-selective biological neuron, overcome some of the common difficulties of image classification: shift, scale and distortion invariance.

2.5.1.1 1-D Convolutional

The one-dimensional CNN for NLP task involved applying a filter window (kernel) over each word in the sentence to extract the n-gram features for different positions. Let $x \in \mathbb{R}^{s \times d}$ be the input sentence matrix, where s is the length of the sentence, and d is the dimension of the embedding used to represent each word in the sentence. Let also denote k be the length of each kernel, where c vector is the kernel for each convolution operation $c \in \mathbb{R}^{kd}$. For each j-th position in the sentence, $\mathbf{w}_j$ is the concatenation of k consecutive words *i.e.*: $\mathbf{w}_j = [x_j, x_{j+1}, \ldots, x_{j+k-1}]$. The extracted one-dimensional feature m_j for each window vector $\mathbf{w}_j$ is computed by applying non-linear function f to the dot product of the window $\mathbf{w}_j$ by the kernel c, plus a bias b:

$$m_j = f\left(\mathbf{w}_j \cdot \mathbf{c} + b\right) \tag{2.25}$$

The non-linear activation function f can take any form, but is most often a hyperbolic tangent or a Rectified Linear Unit (ReLU) [Nair and Hinton 2010]. Figure 2.6 shows two feature maps m with 3-gram and 4-gram $k$ernels.

Channels. In computer vision, images can have several channels (*e.g.* RGB channels). Each image is represented as a combination of pixels with Red, Green and Blue colors intensity at a particular point. In particular, applying a 2-D convolution to an image with different sets of filters and then combing them into a single vector means combining a different view of the image. Each matrix or *view* is referred to as a channel. However, in the case of text, multiple channels may translate into a different representation of the same input text, such as different word vectors for a word. Multi-channel embedding can be either static or non-static (trainable).

2.5.2 Learning Semantic Similarity with CNN

In this section, we describe different works that learn semantic similarity via convolution based architecture. A convolution neural network is designed to extract and identify a local feature in a large structure and combine it to produce a vector representation of a fixed size of that structure, abstracting the most important and informative aspects from that structure for predicting task. For text, the 1D-convolution, as described above, captures an n-gram from a sequence. Kim (2014) apply simple convolution with a pooling layer for sentiment analysis classification. Also, Kalchbrenner et al. (2014) presented the Dynamic Convolutional Neural Network (DCNN) for movie review sentiment prediction. Dynamic convolution is able to capture short and long relations in the sentence. It uses a feature graph to obtain a different size of words. Conneau et al. (2016) propose a very deep CNN (VDCNN) (29 convolutional layers), inspired by computer vision VGG based architecture. The VDCNN works at the character level and uses a small convolution layer followed by a pooling operation.

For the semantic similarity task, He et al. (2015) used various convolution and pooling to extract a stream of tokens. The model consists of two models, sentence and similarity measurement layers that they compare sentence representations using multiple similarity metrics. Yin et al. (2016) introduce attention based CNN for modeling sentence similarity, such as answer selection, paraphrase identification, and textual entailment. They proposed three models 1) attention impact the convolution, 2) attention influences the pooling layer, and finally, 3) a combined model.

Next, we discuss in more detail two different approaches to learning semantic similarity with convolutional based architecture: first, a model proposed by [Severyn and Moschitti 2015], for information retrieval tasks, in particular, for reranking query candidate answer pairs using a sentence similarity model. Second,

an approach that can learn not only the similarity but also dissimilarity [Wang et al. 2016b]. These approaches take advantage of the strong association between similarity and dissimilarity to learn better similarity relations.

2.5.2.1 Learning to Rank Short Text Pairs

ConvNets is a CNN-based model for matching or learning the similarity between text pairs [Severyn and Moschitti 2015]. ConvNets can map inputs, pairs sentences and compute their similarity score. These sentence pairs from the ConvNets are represented as vectors x_q as query and x_d as a document. The similarity function [Bordes et al. 2014] between them is computed as follows:

$$sim\left(x_q, x_d\right) = x_q^T M x_d \tag{2.26}$$

where $M \in \mathbb{R}^{d \times d}$ is a similarity matrix. The objective from the similarity function is to transform the candidate document $\mathbf{x}'_d = M\mathbf{x}_d$ to the closest x_q. The M similarity matrix is learned during the training. After each convolution, there are two additional layers 1) a hidden layer and 2) a softmax. The hidden layer is computed as:

$$\sigma\left(w_h \cdot x + b\right) \tag{2.27}$$

where σ is the non-linearity, a Rectified Linear Unit (ReLU) [Nair and Hinton 2010] that defined as simple $max(0, x)$. The ReLU ensures that all the feature map are positive. w_h is the weight vector and b is the bias. The output of convolutional and pooling layers is a dense factor x that is connected to softmax layers. The softmax layer is computed as a probability distribution over the labels:

$$p(y = j|x) = \frac{e^{x^T \theta_j}}{\sum_{k=1}^{K} e^{x^T \theta_k}} \tag{2.28}$$

where x is the final abstract of the input representation obtained from input layers *i.e.* convolutional and pooling, and θ_k is the weight vector of the k-th class.

In summary, the output of the sentence model, query x_q and document x_d, are the distributional representation. Then, the model learns the similarity matrix M according to Equation 2.26, which produces a similarity score of s_{sim} to capture different aspects of the similarity between the input query x_q and document x_d. The final joint layer of all intermediate vector: x_q^T, x_d^T and similarity score x_{sim} is represented in the signal vector:

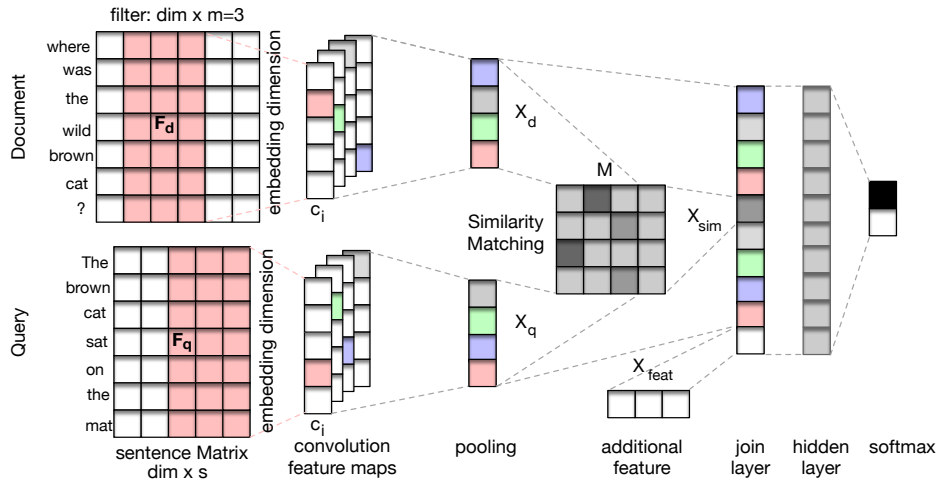

Figure 2.7: Learning to re-rank short text pairs with convolutional [Severyn and Moschitti 2015]. The network learns to optimally represent text pairs and a similarity function in a supervised manner. Figure reproduced from Severyn and Moschitti (2015).

$$x_{\text{join}} = \left[x_q^T; x_{sim}; x_d^T\right] \tag{2.29}$$

The vector is then fed into a fully connected layer that allows for modeling interactions between the joint vector. Finally, the final output is computed with a softmax layer as in Equation 2.28.

2.5.2.2 Sentence Similarity Learning by Lexical De/Composition

Most sentence similarity approaches focus on similarity and ignore the dissimilarity between the two input words or sentences. Wang et al. (2016b) present a CNN based model that takes into account both similar and dissimilar words through the lexical-semantic composing and decomposing the sentences. In particular, the model computes semantic similarity based on word-to-word matching in other sentences (*i.e.* matching words in sentence A with sentence B). Then, each word vector is decomposed into similar and dissimilar based on its similarity. A CNN based model is trained to capture similar and dissimilar features. Finally, the similarity is estimated over the composed vectors.

The model takes a pair of sentences S and T and computes the semantic similarity score $sim(S, T)$. The model uses pre-trained word embedding to have an effective way to represents each word with a distributed vector. As in word embedding words appearing in a similar context tend to have similar meanings. They

build the sentence matrices from $S = [s_1, \ldots, s_i, \ldots, s_m]$ and $T = [t_1, \ldots, t_j, \ldots, t_n]$, where s_i and t_j are the dimension d vectors of the corresponding words and n and m are the sentence lengths of T and S respectively.

In order to learn the similarity between the two sentences, they compare the word coverage, word by word, on the other sentence, and vice versa. As shown in Table 2.3, the E_1 paraphrase *irrelevant* matches the correct paraphrase in E_2 *no related*. Concretely, they consider each word as a semantic unit (primitive unit), and then compute the semantic matching $\hat{S}_i$ for each word in the sentence s_i, by composing full or part of the word vector in the other sentence pair T. For this, the word can match a word s_i in the other phrase or word in, the other sentence, T, and vice versa. The semantic matching function can be computed as follows:

$$\begin{aligned} \hat{s}_i &= f_{\text{match}}\left(s_i, T\right) \quad \forall s_i \in S \\ \hat{t}_j &= f_{\text{match}}\left(t_j, S\right) \quad \forall t_j \in T \end{aligned} \tag{2.30}$$

where $\hat{t}_i$, $\hat{d}_i$ are the semantic matching vectors and f_{match} is the cosine similarity. After the semantic similarity matching phase, the resulting vectors are $\hat{s}_i$ and $\hat{t}_i$. $\hat{s}_i$ or $\hat{t}_i$ is considered to be the word semantic coverage of s_i or t_i. For instance, as in Table 2.3 the word in E_2 *salmon* matched E_1 *sockeye*, which is red *salmon*. The model decomposes the word s_i or t_i based on the similarity matching $\hat{s}_i$ or $\hat{t}_i$ into two components s_i^+ or t_i^+ and the dissimilar part (*red* salmon) s_i^- or t_i^-. The decomposition function is defined as:

$$\begin{aligned} \left[s_i^+; s_i^-\right] &= f_{\text{decomp}}\left(s_i, \hat{s}_i\right) \quad \forall s_i \in S \\ \left[t_j^+; t_j^-\right] &= f_{\text{decomp}}\left(t_j, \hat{t_j}\right) \quad \forall t_j \in T \end{aligned} \tag{2.31}$$

After having both similar and dissimilar component matrix: 1) similar matrix $S^+ = \left[s_1^+, \ldots, s_m^+\right]$ or $T^+ = \left[t_1^+, \ldots, t_m^+\right]$ and 2) dissimilar matrix $S^- = \left[s_1^-, \ldots, s_m^-\right]$ or $T^- = \left[t_1^-, \ldots, s_m^-\right]$. The goal is to use this information, since the similar and dissimilar have a strong relation. For instance, as shown in Table 2.3 it is very difficult to distinguish between E_4 and E_5 which is more similar to E_3. However, after considering both similar and dissimilar, the model can identify that E_3 and E_5 are similar. The model composes both the similar and dissimilar component matrix into a feature vector as follows:

$$\begin{aligned} \vec{S} &= f_{comp}\left(S^+, S^-\right) \\ \vec{T} &= f_{comp}\left(T^+, T^-\right) \end{aligned} \tag{2.32}$$

Finally, the concatenation between the two vectors $\vec{T}$ and $\vec{S}$ and final semantic score prediction.

Table 2.3: Examples from sentence similarity learning by lexical decomposition and composition [Wang et al. 2016b]. The yellow and red colors reflect the similarity and dissimilarity, respectively.

E_1	The research is [irrelevant to sockeye]	red salmon
E_2	The study is [no related] to salmon	-
E_3	The research is relevant to salmon	-
E_4	The study is relevant to sockeye, instead of coho	silver salmon
E_5	The study is relevant to sockeye, rather then flounder	flatfish

$$\text{sim}(S,T) = f_{\text{sim}}(\vec{S},\vec{T}) \tag{2.33}$$

2.5.3 Long Short Term Memory

Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] is able to capture historical information from sequential input sequences. The architecture can handle sequential information as the current input x_t can access the previous output, hidden layer h_{t-1}, at each time step.

The advantages of LSTMs over standard Recurrent Neural Networks (RNN) relies on R **gates** that control the output of each time step, as a function of the previous/old hidden state h_{t-1} and the current time step input x_t: 1) forget gate f_t, 2) input gate i_t, and 3) output gate f_t. These gates can control (update, reject) the memory cell c_t and the hidden state h_t. The LSTM transition function is defined as:

$$\begin{aligned}
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \\
q_t &= \tanh\left(W_q \cdot [h_{t-1}, x_t] + b_q\right) \\
o_t &= \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot q_t \\
h_t &= o_t \odot \tanh\left(c_t\right)
\end{aligned} \tag{2.34}$$

Where σ is a logistic sigmoid function [0,1] and *tanh* a hyperbolic tangent function [-1,1] and $\odot$ denotes an element-wise multiplication. f_t is the function that controls the information from the old memory cell c_t (reject), i_t and q_t are the function to control how much information is stored in the current memory cell c_t and o_t is the function to controls the output c_t.

2.5.4 Learning Semantic Similarity with LSTM

LSTMs [Hochreiter and Schmidhuber 1997] have been used successfully in many NLP tasks, such as text classification [Liu et al. 2016], language modeling [Peters et al. 2018] and machine translation [Bahdanau et al. 2014].

For learning semantic similarity, Wang et al. (2017) propose an LSTM model called the bilateral multi-perspective matching model (BiMPM). The BiLSTM model takes two sentences (P and Q) and encodes them into two directions P against Q and Q against P. Then another BiLSTM is used to aggregate the result (similarity matching) into a matching vector. Finally, the matching vector is used for the final decision with a dense layer. Another approach introduces a two subnetwork, LSTM based, to learn semantic similarity [Chen et al. 2019]. They propose a generative model that relies on two latent variables: 1) the first one representing the syntax and 2) the second one representing the semantics. The model is trained with multiple losses that exploit the alignment of both sentences and word order information.

Next, we discuss in more detail two different approaches that use LSTMs to learn semantic similarity.

2.5.4.1 Manhattan LSTM

LSTM architecture is naturally suited for learning variable-length sequences like a sentence. Mueller and Thyagarajan (2016) propose a Siamese Network based on LSTM to estimate the degree of similarity measure between two sentences. The model is divided into two LSTM network LSTM_1 and LSTM_2, in which each model processes one sentence in a given pair. The model is based on a siamese[3] architecture that relies upon weight sharing technique (*i.e.* $\text{LSTM}_1 = \text{LSTM}_2$) for comparing between two instances. Each LSTM learns to map a variable length of sequence of d-dimensional vector into representation $\mathbb{R}^{d_{rep}}$. Each sentence is represented as a sentence word vector $x_1, ..x_T$ (*i.e.* pre-trained word embedding, which we described in Section 2.4.1) is passed into LSTM encoder as shown in Equations 2.34. The final encoding of representation for each sentence is encoded by the last hidden state $h_T \in \mathbb{R}^{d_{rep}}$. For each given sentence pair they apply a similarity function:

$$g : \mathbb{R}^{d_{rep}} \times \mathbb{R}^{d_{rep}} \rightarrow \mathbb{R} \tag{2.35}$$

[3]Siamese networks seem to perform well on similarity tasks and have been used for tasks like comparing two objects in computer vision task [Koch et al. 2015].

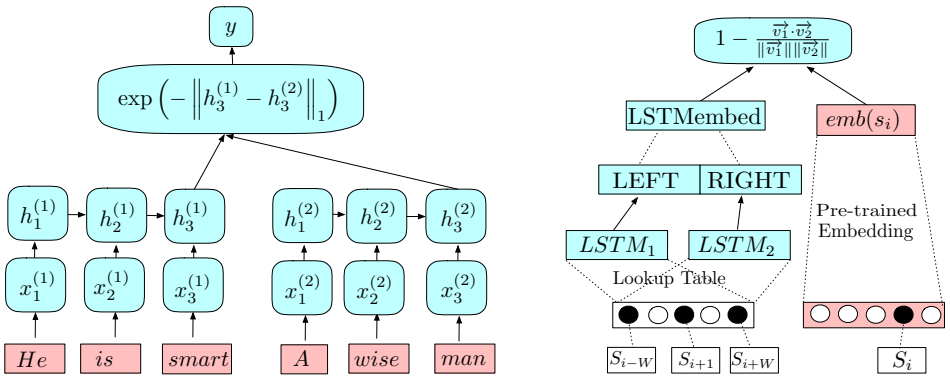

Figure 2.8: Example of learning the semantic similarity with LSTM. Learning the semantic similarity with Manhaten distance (Left) [Mueller and Thyagarajan 2016]. (Right) Knowledge based embedding with LSTM (LSTMEmbed) [Iacobacci 2019]. Figures reproduce from Mueller and Thyagarajan (2016), Iacobacci (2019).

where g is the similarity function. For each given pair this similarity function is applied to their encoder-representation.

Unlike the standard LSTM that is used in Language modeling, which predicts the next word from the previous one, this LSTM is a simple encoder [Sutskever et al. 2014], as shown in Figure 2.8 (Left). The LSTM encoder is trained to learn the similarity between the two sentence representations $h_{T_1}^{(1)}, h_{T_2}^{(2)}$ and predicts the similarity. The similarity function is the Manhattan distance:

$$\left(h_{T_1}^{(1)}, h_{T_2}^{(2)}\right) = \exp\left(-\left\|h_{T_1}^{(1)} - h_{T_2}^{(2)}\right\|_1\right) \in [0, 1] \tag{2.36}$$

According to the author, the Manhattan distance g as in Equations 2.35, outperforms the most common approaches, such as cosine similarity [Yih et al. 2011].

2.5.4.2 LSTMEmbed

Most recent approaches learn the embedding with feed-forward neural (FFN) networks. However, the advantage of LSTMs over FFN is that it takes into consideration both the word context and word-order awareness. Iacobacci (2019) presented LSTMEmbed, a bidirectional LSTM (BiLSTM) based model to learn the knowledge based word embedding. They use the tagged sense to provide the input context in both directions 1) preceding context $s_i - W, \ldots, s_{i-1}$ and 2) posterior context $s_{i+1}, \ldots, s_{i+W}$, where s_j, $(j \in [i - W, \ldots, i + W])$ is the word sense from, external knowledge, an existing inventory Bablenet [Navigli and Ponzetto 2012]. Each token is associated with an embedding vector $\mathbf{v}(s_j) \in \mathbb{R}^n$, in a shared look-up ta-

ble. The BiLSTM reads the sequences (preceding context, posterior context) in both direction (*i.e.* left-to-right and right-to-left):

$$\begin{aligned} o_l &= \mathrm{lstm}_l\left(\mathbf{v}\left(s_{i-}w\right), \ldots, \mathbf{v}\left(s_{i-1}\right)\right) \\ o_r &= \mathrm{lstm}_r\left(\mathbf{v}\left(s_{i+1}\right), \ldots, \mathbf{v}\left(s_{i+W}\right)\right) \end{aligned} \tag{2.37}$$

Next, the LSTMs are merged and projected linearly via a fully connected or dense layer:

$$\text{out}_{\,LSTMEmbed} = \mathbf{W}^o\left(o_l \oplus o_r\right) \tag{2.38}$$

where $\mathbf{W}^o \in \mathbb{R}^{2m \times m}$ is the weight matrix with m as the LSTM dimension. The model $out_{LSTMEmbed}$ is compared with a pre-trained model vector $emb(s_i)$ such as GloVe or word2vec, that we described in Section 2.4.2. The model is trained to maximize the similarity between $out_{LSTMEmbed}$ and $emb(s_i)$. Therefore, the loss function is a similarity distance, cosine similarity:

$$loss = 1 - \mathcal{S}(\overrightarrow{v_1}, \overrightarrow{v_2}) = 1 - \frac{\overrightarrow{v_1} \cdot \overrightarrow{v_2}}{\|\overrightarrow{v_1}\| \|\overrightarrow{v_2}\|} \tag{2.39}$$

After the training, the model obtains joint representations in the same vector space from the look-up table. Precisely, senses and latent semantic representations of words are jointed in the same vector space.

2.5.5 Attention Mechanism

Attention-based models have shown a promising results in various NLP tasks, such as machine translation [Bahdanau et al. 2014], and caption generation [Xu et al. 2015]. Such a mechanism learns to focus attention on a specific part of the input (*e.g.* target word in a sentence). The basic concept of the attention-based encoder-decoder can be described as follows:

First, the **encoder** is used to process and encode the input sentence into a context vector (last hidden state). The concept of attention in this scenario is the summarization of the input sentence. In other words, all initial hidden states are ignored, and only the final state will be considered as the initial state to the decoder.

Second, the **decoder** generates the word or the summarization of words in that sentence, from the context vector.

However, one of the drawbacks is that the decoder depends on the fixed-length context vector to generate the output, which is not practical in long sentences (it has often overlooked the first element once it completes the entire sequence). For

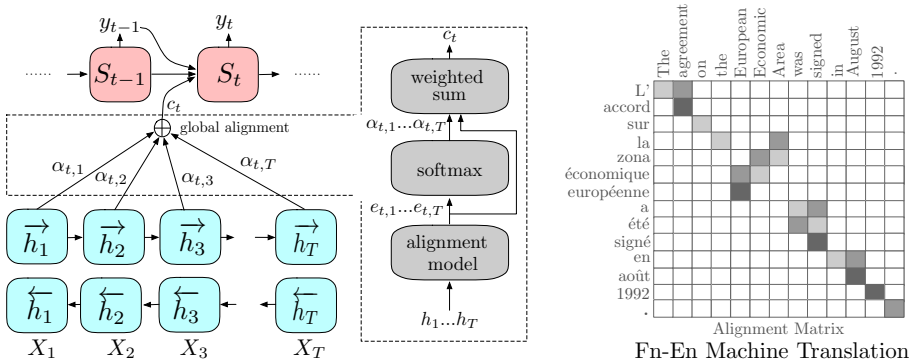

Figure 2.9: (Left) A word alignment model, dot-line-box, that generate context c_t for the target word y_t from a source sentence $X_1.., X_T$. (Right) Alignment matrix correlation score between the input and target words in French-to-English machine translation scenario. Figures adopted from Bahdanau et al. (2014).

instance, in machine translation, a wrong context will lead to incorrect translation. To solve this problem, rather than generating a single fixed-length context vector from the last hidden state of the encoder, the attention creates shortcuts between the entire source data and the context vector [Bahdanau et al. 2014]. The shortcut of these weights is adjustable for each output. The proposed model considers not only the context vector but also the relative importance in the sequences. The context vector access the full sequence, and the model learns the alignments between the target and the input source. The context vector depends on three information, as shown in Figure 2.9 (Left) hidden states from encoder and decoder and the alignment between the input source and the target.

The model introduced by Bahdanau et al. (2014) use a BiLSTM as encoder to generate sequences for each sentence, in both direction (h_1, h_2). The result vector h_1 and h_2 are then concatenated as forward and backward as follows:

$$\boldsymbol{h}_j = \left[\overrightarrow{\boldsymbol{h}}_j^\top; \overleftarrow{\boldsymbol{h}}_j^\top\right]^\top, j = 1, \ldots, T \tag{2.40}$$

The decoder take the hidden state $\boldsymbol{s}_t = f\left(\boldsymbol{s}_{t-1}, y_{t-1}, \mathbf{c}_i\right)$ for a word at position t, $(t = 1, ...m,)$ where c_t is the sum of hidden state of the input sequences weighted by alignment scores:

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \tag{2.41}$$

where α_{tj} is the weight computed at each time t step for hidden state h_j, T is the number of time steps for the input sequence. The c_t context vector is used to com-

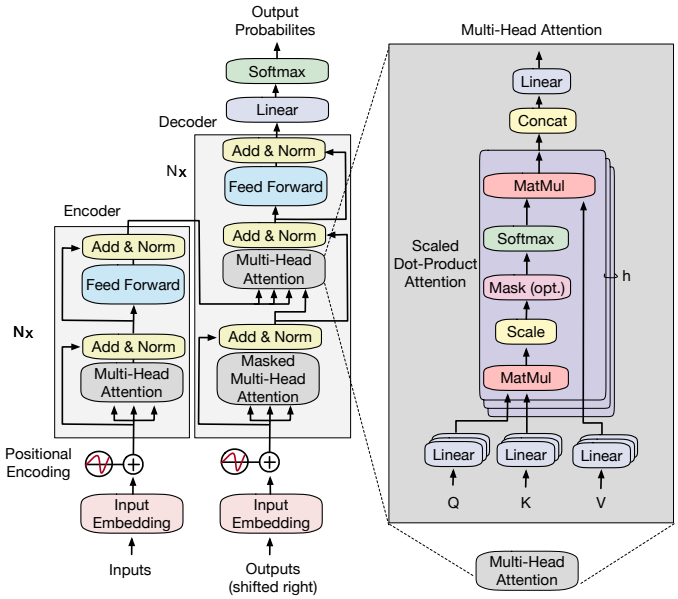

Figure 2.10: Full architecture of the transformer. Figure reproduced from Vaswani et al. (2017).

pute the new state sequence s, where s_t depends on previous state s_{t-1}. The α_{tj} weight are then computed as:

$$\alpha_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{T} \exp\left(e_{tk}\right)} \tag{2.42}$$

The alignment score a is computed as feed forward network, and the alignment model score is calculated by $e_{tj} = a\left(s_{t-1}, h_j\right)$, for the inputs/output at j position.

Next, we discuss the Transformer architecture or self-attention model without any recurrent network, which offers a significant improvement over the methods we described in this section, such as attention-based aligned recurrent models.

2.5.6 Transformer

In the previous section, we described the best ways to capture dependencies in long sequences, LSTM with attention, in particular, the encoder-decoder based architecture called the Seq2Seq model. The encoder maps the input sequence into a higher dimensional space. Then, that abstract vector is fed into the decoder, which turns it into the output sequence.

However, although these models obtain state-of-the-art performance in sequence modeling (*i.e.* language modeling [Sutskever et al. 2014] and machine translation

[Bahdanau et al. 2014]). recurrent networks (*i.e.* GRU, LSTM, etc.) have some drawbacks: 1) they slow down the training (difficult to parallelize) and 2) are computationally expensive.

Vaswani et al. (2017) introduce Transformer architecture, which is able to deal with Encoder and Decoder without any Recurrent Networks. Transformer architecture is based entirely on self-attention without any aligned recurrent network, as we described above in the attention Section 5.2.4.

As shown in Figure 2.10 the transformer uses scaled dot product attention. The weight is determined by the dot-product as follows:

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.43}$$

where the input is a set of pair **Key** K and **value** V of a dimension n, and $\frac{1}{\sqrt{d_k}}$ is the scaling factor. In the encoder, both K and V are the hidden states. The decoder compresses previous output as query Q with dimension m and produce the next output by mapping Q and the set of the Ks and Vs.

The main concept of Multi-Head Attention (MHA) is that rather than computing the attention once, the MHA runs through the scaled dot product attention many times in parallel. As shown in Figure 2.10 the independent outputs of attention are concatenated and linearly reshaped into the required dimension.

$$\begin{aligned}\text{MultiHead}\,(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}\,[\,\text{head}_{\,1}; \ldots;\, \text{head}_{\,h}]\,\mathbf{W}^O \\ \text{where head}_{\,i} &= \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right)\end{aligned} \tag{2.44}$$

where $\mathbf{W}_i^{Q,K,V}$ and $\mathbf{W}^O$ are matrices for parameters to be learned by the network.

The encoder, the left hand block in Figure 2.10, produces an attention-based representation with the ability to locate or find a particular piece of information in a large context. They stack 6 identical layers $N_X = 6$, and each layer has 1) multi-head attention layers and 2) a fully connected layer. Each sub-layer has a skip connection (residual) connection and normalization layers.

The decoder also has $N_X = 6$ identical layers. Each layer has two sub-layers of attention (multi-head) and a fully connected layer. The multi-head attention of the input (first attention in the decoder) is modified with masking to block attending the future token while calculating the current position. The rest of the layers are similar to the encoder that utilized skip connection and normalization layers.

Finally, at the top of each decoder, a softmax and linear layer is added for the final output. A positional encoding with sinusoid-wave is introduced to preserve

the position information. The positional encoding can be added directly to the input, as it has the same dimension as the input embedding.

2.5.7 Learning Semantic Similarity with Transformer

In this section, we offer a simple introduction to a Universal Encoder that uses a stack of transformers to learn semantics similarity. In Section 2.6.2, we discuss BERT [Devlin et al. 2019] the state-of-the-art model in semantic similarity based on the Transformer.

2.5.7.1 Universal Sentence Encoder

The Universal Sentence Encoder [Cer et al. 2018] is a transformer-based model that encodes sentences into a sentence embedding vector. The model uses transformer-based architecture to construct the sentence embedding with an encoding sub-graph. The sub-graph utilizes the attention mechanism to compute the context of words in a sentence. These context-aware representations take into account word order to identify all of the other words in the sentence. These representations are mapped into an encoding vector with a fixed length by computing the sum (element-wise) of the representation at each word position. The encoder takes a lower-case tokenized PTB string as its input and generates a 512-dimensional vector as the embedded sentence.

The model is designed to be universal for general use for many tasks. This is achieved by introducing multi-task learning, a single encoding, to be able to feed multiple down-stream tasks. For instance, one of the supported tasks is the Skip-Though like application [Kiros et al. 2015], in which the Transformer replaces the LSTM for classification tasks over supervised data.

The data used to train the sentence encoders in an unsupervised way are extracted from a variety of web resources, such as Wikipedia, question-answer pages, discussion forums, and web news. They improve unsupervised learning with supervised data training with data augmentation techniques.

For transfer learning tasks such as sentence similarity, similarity can be computed directly from two-sentence embedding with cosine similarity:

$$\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \left(1 - \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\right)/\pi\right) \tag{2.45}$$

where $\mathrm{sim}(\mathbf{u}, \mathbf{v})$ is similarity based on angular distance. Angular similarity distinguishes near similarity vectors with small changes better than raw cosine.

Most current state-of-the-art approaches in many tasks in NLP are based on transformer architecture, such as BERT [Devlin et al. 2019] and GPT-2 [Radford et al. 2019]. Unlike word embedding, which is described in Section 2.4.2, the transformer-based model, pre-trained language modeling, is able to learn dynamic embeddings. In the next section we explore this dynamic embedding, which is called Contextual Word Embedding Learning.

2.6 Contextual Word Embedding Learning

All word embedding approaches that we discussed in Section 2.4.2, are context insensitive or context independent *i.e.* each word can have only one vector representation.

- Each word always has the same vector representation, regardless of the context in which sentence or tokens occur. Especially for a word with more than one meaning or sense, a polysemic word, one vector representation is not enough to encompass its different meanings.
- Even when a word that has only one meaning or sense, its occurrences still have different semantic aspects, such as syntactic behavior.

Some research directions have proposed the injection of senses (meanings of the word in a different context) into the word embedding semantic space [Mancini et al. 2017, Iacobacci 2019]. However, these approaches rely on a large annotated corpus based on external resources, such as Babelnet [Navigli and Ponzetto 2012] and Wordnet [Miller 1998], which make it unpractical in a real application. Also, despite their impressive performance over standard word embedding [Mikolov et al. 2013b], these approaches are bounded by word2vec architecture limitations as they are context-independent.

To solve this, dynamic word embeddings were introduced. Contextual word embedding learning is based on the context for each token rather than a static context-independent embedding. Next, we present two methods that led to a breakthrough in the field, with many NLP tasks overperforming humans in Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al. 2018].

Table 2.4: Example of Nearest neighbors from [Peters et al. 2018] to highly polysemous word "play" using static embedding GloVe [Pennington et al. 2014] and the context embeddings from a biLM.

Source	Nearest Neighbor
GloVe: play	playing, game, games, played, players, plays, player, **play**, football, multiplayer
BiLM: Chico Ruiz made a spectacular **play** on on Alusik's	Kieffer, the only junior in the group, was a commended for his ability to hit in the clutch as well as his all-round excellent **play**.

2.6.1 ELMo

The Embeddings from Language Models (ELMo) [Peters et al. 2018] is an embedding that derived from the BiLSTM based language model. The ELMo uses a two layers bi-directional language model (biLM). For each computed token t_k by BiLSTM, the biLM computes a set of representations:

$$\begin{aligned} R_k &= \left\{ \mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1, \ldots, L \right\} \\ &= \left\{ \mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L \right\} \end{aligned} \tag{2.46}$$

where L represents the number of layers in biLM, $\mathbf{x}_k^{LM}$ is the context-independent token, $\mathbf{h}_{k,j}^{LM} = \left[\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM} \right]$ are the top layer BiLSTM.

Table 2.4 shows that the BiLM is able to disambiguate the correct sense of the word in the source sense. On the other hand, GloVe vector treats the word as a different part of speech (*i.e.* game as nouns and "playing" as verbs), but the sense of the word is concentrated in the sport-related "Play". This example shows the effectiveness of contextualized representation, which could be applied to a variety of NLP tasks.

2.6.2 BERT

Bidirectional Encoder Representations from Transformers or BERT [Devlin et al. 2019] is another contextual language model like ELMo; however, In contrast to ELMo, it does not rely on recurrent language models with static word embedding initialization, but provides an end-to-end language model that is based entirely on contextualized token embeddings. For this, a transformer [Vaswani et al. 2017] based architecture is used in combination with masked language modeling targets that allow for training a model that can see the context from left and right handed

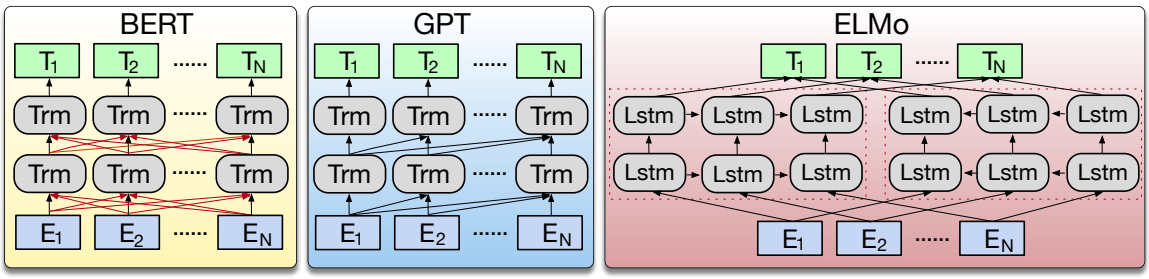

Figure 2.11: Different architecture of pretraining models. GPT uses a transformer with a left-to-right approach. BERT uses a Transformer with a bi-directional based model that condition to the right and left context in all layers. Elmo uses a Bilstm with the concatenation of both LSTMs (right-to-left and left-to-right) to generate features. ELMo is the only feature-based method, while BERT and GPT are fine-tuned approaches. Figure reproduced from Devlin et al. (2019).

perspective at the same time. The idea of self-attention in the transformer, and non-directional language modeling results in extraordinary performance gains compared to previous approaches.

The Masked Language Model uses the output of the masked word's position to predict the masked word. The Masking covers 15% of words in the input and tries to predict the masked word:

Input `[CLS] Rosa attended the play at the firework festival.`

Randomly Mask `[CLS] Rosa did not want to [MASK] card-game at the festival.`

Unlike the GPT [Radford et al. 2018], which uses a left-to-right transformer, the transformer encoder in BERT handles the entire sequence from left-to-right or right-to-left simultaneously, as shown (red connections) in Figure 2.11. Therefore, although this is described as bidirectional, it could be regarded as a non-directional encoding. BERT has shown groundbreaking results in many tasks such as question answering[4] and Natural Language Inference (NLI). However, according to its main authors, it is not suited for the Semantic Textual Similarity (STS) task, since it does not generate a meaningful vector to compute the cosine distance. The most common approach to averaging the BERT output layer is called BERT embedding. However, the result is worse [5] than static embedding, such as GloVe [Pennington

[4]The input for BERT for sentence pair consists of two sentences separated by [*SEP*] token.
[5]Our finding and confirmed by [Reimers and Gurevych 2019]

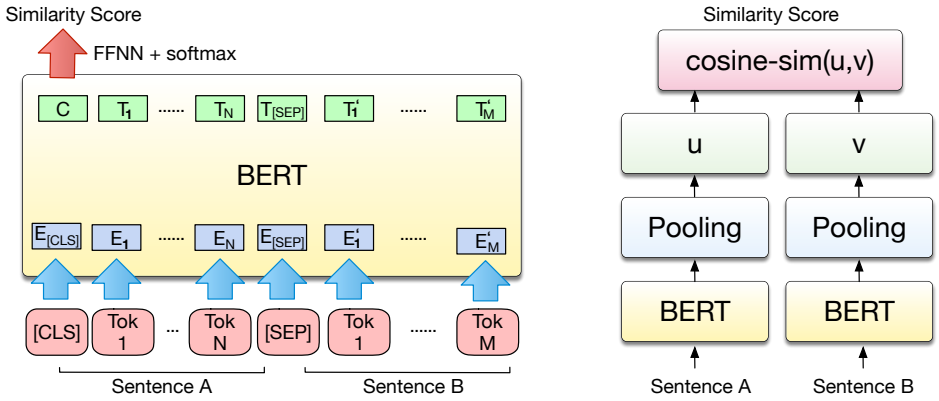

Figure 2.12: Learning the semantic similarity with BERT [Devlin et al. 2019]. Left) Fine-tuning the model with one FFNN layer with softmax. Right) adding pooling operation to derive fixed sentence embedding and compute the mean of all vectors [Reimers and Gurevych 2019]. Figures reproduce from Devlin et al. (2019), Reimers and Gurevych (2019).

et al. 2014]. There two most common approaches to computing semantic similarity with BERT as shown in Figure 2.12, Left) fine-tuning the model with one FFNN layer with softmax Right) are adding pooling operations to derive fixed sentence embedding and computing the mean of all vectors [Reimers and Gurevych 2019]. In this thesis, we follow the fine-tuning approach since it yields better results and is more robust in fine-tuning procedures for different down-stream tasks.

Chapter 3

Textual Visual Semantic Dataset for Text Spotting

The relation between text and its surrounding environment is very important to understanding text in the scene. While there are some publicly available datasets for text spotting, none includes information about visual context in the image. Therefore, in this chapter, we introduce a textual visual semantic context dataset, containing scene information, such as objects or locations, for text spotting tasks. Our goal is to fill this gap, providing a closer vision and language by understanding the scene text and its environmental visual context.

In this chapter, we describe the datasets used in this thesis. Since the focus of this work is simply enhancing scene recognition by adding visual context information to the text spotting system, the same datasets for training and evaluation are used in the entire thesis, and so we introduce and describe them in this chapter.

There are several existing publicly available datasets for text spotting (text detection and recognition), which we describe in Section 3.2. However, we also make our contribution to this field by introducing visual context information to the biggest pre-existing dataset, COCO-text. Section 3.3 describes our proposal for textual visual context generations. In Section 3.4, we show how we can construct a visual context for pre-existing datasets such as COCO-text, which can be used with any existing dataset.

3.1 Summary of Contribution

In this chapter, we describe in depth the construction of the visual context dataset. This dataset is based on the COCO-text [Veit et al. 2016], which uses the Microsoft COCO dataset [Lin et al. 2014] and annotates texts appearing in the images. We

Table 3.1: A various end-to-end text recognition (a.k.a text spotting) dataset.

End-to-End Text Recognition Dataset			
Label	Description	Dictionary	# images
IC03	ICDAR 2003 test dataset [Lucas et al. 2003]	-	251
IC03-Full	ICDAR 2003 test dataset with fixed lexicon [Lucas et al. 2003]	860	251
SVT	Street view test dataset [Wang and Belongie 2010]	-	249
SVT-50	Street view dataset with fixed lexicon [Wang and Belongie 2010]	50	249
IC13	ICDAR 2013 test dataset [Karatzas et al. 2013]	-	233
IC17-T3	ICDAR 2017 Task-3 COCO-text dataset [Gomez et al. 2017]	-	145k

Table 3.2: A various word recognition dataset. The images are cropped word images for recognizing task only.

Text Recognition Dataset			
Label	Description	Dictionary	# images
Synth90k	Synthetic dataset with 90k dict [Jaderberg et al. 2014b]	90k	900k
SynthRand	Synthetic dataset with random text [Jaderberg et al. 2014b]	-	900k
SVT	Street view test dataset [Wang and Belongie 2010]	-	647
IC13	ICDAR 2013 test dataset [Karatzas et al. 2013]	-	1k
IC17-T2	ICDAR 2017 Task-2 COCO-text dataset [Gomez et al. 2017]	-	10k

further extend the dataset by using out-of-the-box tools to extract visual context or additional information from images. Usually the computer vision community tackles this problem by dividing the task into two sub-models, one for text and other for object [Zhu et al. 2016, Kang et al. 2017, Prasad and Wai Kin Kong 2018].

However, our main contribution is this combined visual context dataset that provides the unrestricted-OCR research community the chance to use semantic relatedness between text and image to improve the results.

3.2 Publicly Available Datasets

In this section we describe various publicly available text spotting datasets.

3.2.1 Synthetic Datasets

As shown in Table 3.2 datasets containing real-word images have sizes in the range of at most a few thousand, with a very limited vocabulary. Therefore, [Jaderberg et al. 2014b] introduced a synthetic data generator without any human label cost. All current state-of-the-art approaches are trained on this datasets [Ghosh et al. 2017, Shi et al. 2016, Jaderberg et al. 2016, Jaderberg, Shi et al. 2017, Gao et al. 2017, Fang et al. 2018].

Figure 3.1: A text spotting data sample from (a) COCO-text, (b) Google street view (SVT) showing the bounding box in green (annotation). (c) An example from the ICDAR 2003 recognition test dataset.

Figure 3.2: A random example from the synthetic dataset [Jaderberg et al. 2014b]. The only big dataset that is currently publicly available for training deep learning model.

Synth90k. This dataset is a synthetically generated dataset that consists of 9 million word images with a 90k word dictionary [Jaderberg et al. 2014b]. The 90k dictionary consists of an English word form lexicon from the Hunspell open source spell checker. The dictionary includes 50k root words, together with prefixes and suffixes. The author also added words from other datasets (ICDAR, SVT, and IIIT).

SynthRand. The same work introduces another synthetic dataset of word images [Jaderberg et al. 2014b], which consists of 8 million training images and a test set of 900k images. Unlike synth90k, which uses a pre-defined dictionary, the words generated in this case are completely random strings of characters, where the maximum length of the generated characters is ten.

Note that these two datasets are the biggest datasets that are currently publicly available for text spotting.

3.2.2 ICDAR Datasets

ICDAR 2013 (IC13) [Karatzas et al. 2013]. The ICDAR 2013 dataset consists of two datasets 1) text localization and 2) text segmentation. Text localization consists of 328 training and 233 test images. ICDAR 2013 dataset represents the evaluation of scene text understanding tasks: localization, segmentation, and recognition.

ICDAR 2015 (IC15) [Karatzas et al. 2015]. ICDAR 2015 datasets consist of 1000 training samples and 500 test images. All images were taken by a camera in a variety of indoor/outdoor scenarios in a multilingual environment and contain resolution images with an average size of (1280*720) pixels. ICDAR 2015 is more focused on incidental scene text detection and recognition.

ICDAR 2017 (IC17) [Veit et al. 2016]. ICDAR 2017 is based on COCO-text and aims for end-to-end text spotting as in ICDAR17-Task 3 (*i.e.* detection and recognition). The dataset consists of 43,686 full images with 145,859 text instances for training,

and 10,000 images and 27,550 instances for validation. We describe COCO-text in more detail in the following section.

Figure 3.1 shows some text spotting data examples from the ICDAR datasets, current dataset (a) ICDAR 2017, and the first proposed dataset (c) ICDAR 2003.

3.2.3 Street View Text Dataset

This dataset consists of 249 images for training and 100 images for testing downloaded from Google Street View (SVT) [Wang and Belongie 2010]. This is the first dataset that deals with text image in a real scenario, in the wild, with examples such as shop signs in a wide range of fonts and graphic styles. Note that many word text images have not been annotated in this dataset.

3.2.4 COCO-text

This dataset is based on Microsoft COCO [Lin et al. 2014] (Common Objects in Context), and consists of 63,686 images, 173,589 text instance (annotations of the images). The COCO-text dataset differs from the other datasets in three aspects. First, the dataset was not collected with text recognition in mind. Thus, the annotated text instances lie in their natural context. Second, it contains a wide variety of text instances, such as machine-printed and handwritten text. Finally, the COCO-text has a much larger scale than other datasets for text detection and recognition. This is a challenging dataset with a lot of noise, false-positives, and illegible text.

3.3 Textual Visual Semantic Dataset

The primary purpose of the textual visual semantic dataset is to enable learning of the semantic relation between the image and the text that happen to be together in an image, thereby improving recognition accuracy.

We employ recent state-of-the-art tools to extract textual information for each image. In particular, for each image, we obtain the following information: (1) Spotted text candidates (hypotheses provided by existing state-of-the-art text spotting systems) and (2) surrounding visual context information in the image such the location of the objects depicted in it.

Table 3.3: A description of the various word recognition dataset with or without the proposed visual context information.

Text Recognition Dataset				
Label	Description	Dictionary	# word images	text †
IC17-T2	from COCO-text dataset Task-2 [Gomez et al. 2017]	-	46k	-
Synth90k	Synthetic dataset with 90k dict [Jaderberg et al. 2014b]	90k	900k	-
SVT	Street view test dataset [Wang and Belongie 2010]	-	647	-
IC13	ICDAR 2013 test dataset [Karatzas et al. 2013]	-	1k	-
IC17-V-T3	Image+Textual dataset from IC17 Task-3 (ours)	-	10k	25k
COCO-text-V	Image+Textual dataset from COCO-text (ours)	-	16k	60k
COCO-Pairs	Textual dataset from COCO-text (ours)	-	-	158k

3.3.1 Text Hypotheses Extraction

To extract the text associated with each image or bounding box, we employ several pre-trained Text Spotting baselines to generate k text hypotheses. All the pre-trained models are trained on a synthetic dataset [Jaderberg et al. 2014b]:

Convolutional Neural Network-90k-Dictionary [Jaderberg et al. 2016]. The first baseline is a CNN with fixed lexicon-based recognition, able to recognize words in a pre-defined 90k-word dictionary W. Each dictionary entry w corresponds to a potential output in a 90k multi-class classification problem. The dictionary consists of different forms of English words (*e.g.* verb, noun, adjective, etc.). In short, the model classifies each input word-image into a pre-defined word in the 90k fixed lexicon. Each word $w \in W$ in the dictionary corresponds to the one output neuron. The final output $word$ for a given image x is written as:

$$word = \arg\max_{w \in \mathcal{W}} P(w|x, lexicon) \tag{3.1}$$

Convolutional Recurrent Neural Network (CRNN) [Shi et al. 2016]. The second baseline is a CRNN that can learn the words directly from sequence labels, without relying on character annotations. The encoder uses a CNN to extract a set of features from the image. Specifically, the network uses CNN without fully connected layers to extract sequential feature representations from an input image and then feed them into the bidirectional RNN. A Connectionist Temporal Classification [Graves et al. 2006] based method is used to convert the per-frame predictions made by the RNN into a label sequence as follows:

$$\mathbf{I}^* \approx \mathcal{B}\left(\arg\max_{\pi} p(\boldsymbol{\pi}|\mathbf{y})\right) \tag{3.2}$$

Table 3.4: Sample from the dataset. The text hypothesis comes from existing Text Spotting baselines, and the visual context information comes from out-of-the-box computer vision classifiers. **Boldface** fonts reflect the ground truth.

Text hypothesis	Object	Scene	Caption
11, il, j, m, ...	railroad	train	a train is on a train track with a train on it
lossing, docile, dow, **dell**, ...	bookshop	bookstore	a woman sitting at a table with a laptop
29th, 2th, **2011**, zit, ...	parking	shopping	a man is holding a cell phone while standing
happy, hooping, happily, nappy, ...	childs	bib	a cake with a bunch of different types of scissors
coke, gulp, slurp, fluky,...	plate	pizzeria	a table with a pizza and a fork on it
will, **wii**, xviii, wit,....	remote	room	a close up of a remote control on a table

where $\mathcal{B}$ is a sequence-to-sequence mapping function, $\boldsymbol{\pi}$ a sequence label and $\mathbf{I}^*$ the sequence.

LSTM-Visual Attention (LSTM-V) [Ghosh et al. 2017]. The third baseline also generates final output words as probable character sequences, without relying on a lexicon. The network is based on an encoder-decoder architecture with a visual attention mechanism. In particular, they use the pre-trained CNN model as the encoder [Jaderberg et al. 2016] mentioned above, but without the final layer to extract the most important feature vectors from each text image. That feature vector is used to reduce model complexity through the soft attention model [Xu et al. 2015] which focuses on the most relevant parts of the image at every step. The LSTM [Hochreiter and Schmidhuber 1997] decoder computes the output character probability y for each time step and the visual attention α, where L are the deep model output parameters of each layer:

$$P\left(y_t|\alpha, y_{t-1}\right) \sim \exp\left(\mathrm{L}_0\left(Ey_{t-1} + \mathrm{L}_\mathrm{h}h_t + \mathrm{L}_\mathrm{z}\hat{z}_t\right)\right) \tag{3.3}$$

CNN-Attention [Fang et al. 2018]. Finally, we employ the most recent state-of-the-art system, which also produces the final output words as probable character sequences, without a fixed lexicon. The model is based on a CNN encoder-decoder with attention and a CNN character language model. The final character prediction is an element-wise addition of the attention and language vector as:

$$p\left(y_k|y_{k-1}, \ldots, y_1\right) = p_a + p_l \tag{3.4}$$

where p_a and p_l are softmax function that converts both the attention vector and the language vector to the predicted characters separately.

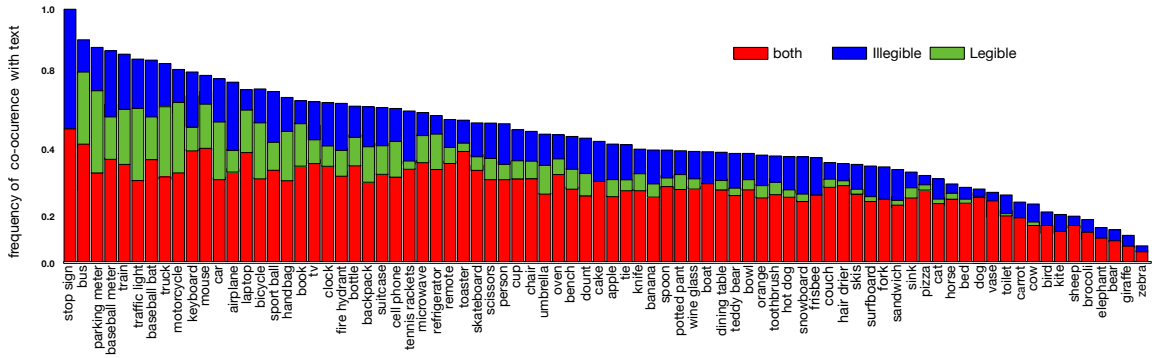

Figure 3.3: The frequency of objects in MS COCO co-occurs with text, as can be seen, that the presence of particular objects is very informative regarding text presence. Especially traffic and sports scenes almost always contain text and nature scenes with animals rarely contain text. Figure reproduce from COCO-text Veit et al. (2016).

3.3.2 Visual Context Information

While there a number of publicly available datasets for text spotting, there is no visual context dataset for this task such as scene information or an image description. We propose to introduce a textual dataset for the same task. This section describes our textual dataset for text spotting. In particular, we will use out-of-the-box state-of-the-art classifiers to extract the image context information. We obtain three kinds of context information: the object present in the image, the location/scenarios seen in the image, and a textual description or caption.

3.3.3 Object Information

To capture the *visual context* from each image, we will use visual object classifiers to extract the image context information that will be used to re-rank candidate words according to their semantic relatedness with the context. The output of the following classifiers is a 1000-dimensional vector with the probabilities of 1000 object types. In particular, we extract the top-5 objects with the highest possibilities.

GoogLeNet [Szegedy et al. 2015]. The design of this network architecture is based on an inception module, which uses 1-D convolution to drastically reduce the number of parameters. Also, a fully connected layer is replaced with a global average pooling at the end of the network. The network consists of 22 layer Deep CNN with a reduced parameter. It has a top-5 error rate of 6.67%.

ResNet [He et al. 2016]. A novel very deep architecture with a residual network (or

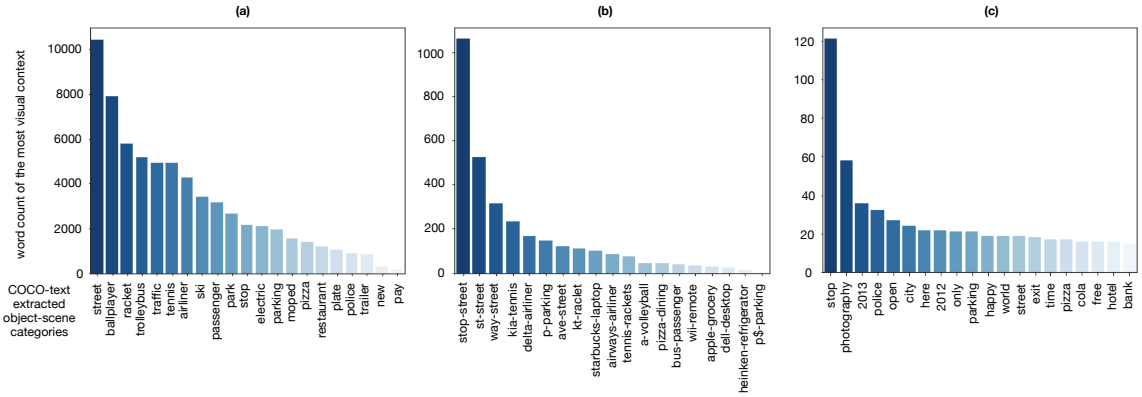

Figure 3.4: (a) Frequency of objects in COCO-text images. (b) Most common pair (text-object) in the training dataset (c) Frequency count of the most visual context in the testing dataset.

shortcut connection) featuring heavy batch normalization. The residual shortcut is also known as gated recurrent units and has a strong resemblance to recent successful elements applied in RNNs. Thanks to this residual connection, they were able to train a deeper model with 152 layers while maintaining a lower complexity than simple VGGNet. The model achieves a top-5 error rate of 3.57%.

Inception-Resnet [Szegedy et al. 2017]. Inspired by the breakthrough ResNet performance, a hybrid-inception module was proposed. In order to add the residual connection to inception modules, the input/output after convolution must have the same dimensions. Also, the pooling inside the inception modules was replaced with a residual connection. Inception-ResNet combines the two architectures (Inception modules and residual connection) to boost performance even further. We use Inception-Resnet-v2, with a top-5 error rate 3.1%.

3.3.4 Scene Information

Additionally, we considered a scene classifier [Zhou et al. 2017] to extract scene information from each image. We used a pre-trained scene classifier *Places365-ResNet*[1] to extract scene categories. According to the authors of *Places365-ResNet* the network achieved good result in *top-5 accuracy*, which make it ideal for multiple visual context extraction. The output from this classifier consists of 365 scene categories. Also, we consider a threshold to extract the most likely classes in the images, and eliminate low confidence predictions.

[1]http://places2.csail.mit.edu/

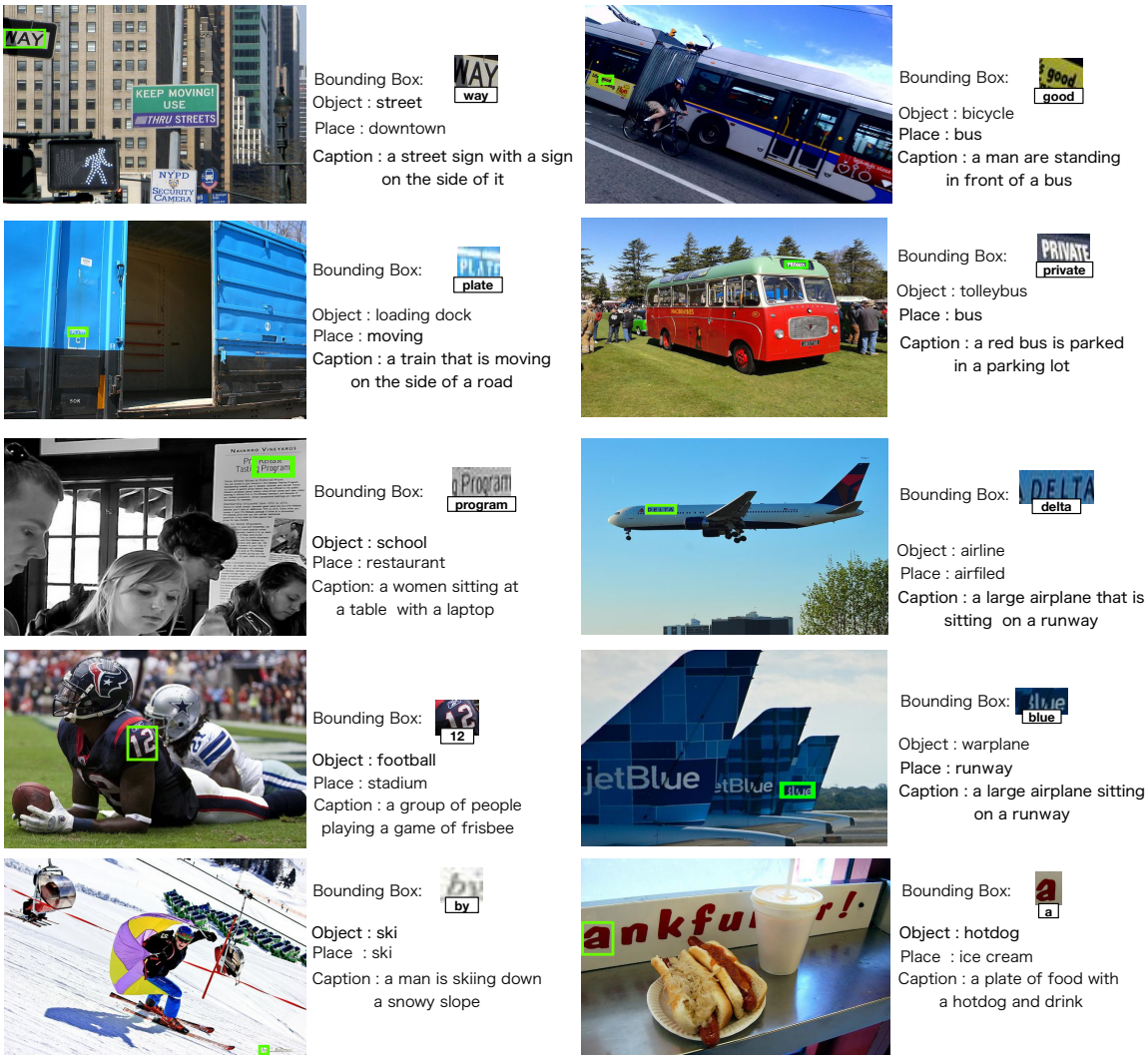

Figure 3.5: Some random examples extracted from COCO-text with the visual context information. For each sample, there are a bounding box, full images and information from the image, such as an object, scene scenario, and image description.

3.3.5 Image Description

Finally, we use a caption generator to extract more visual context information from each image as a natural language description. Image caption generation approaches can use either top-down or bottom-up approaches. The bottom-up approach consists of detecting objects in the image and then attempting to combine the identified objects into a caption [Karpathy and Fei-Fei 2015]. On the other hand, the top-down approach learns the semantic representation of the image, which is then decoded into the caption [Vinyals et al. 2015]. Most recently, the combination of both approaches [Anderson et al. 2018] has achieved state-of-the-art results in many tasks such as Visual Question Answering. Most current state-of-the-art systems adopt the *top-down* approach using RNN-based architectures. In this work, we use a top-down model to extract the visual description of the image.

As we are more interested in the visual context than the quality of the generated sequence, we use this standard architecture [Vinyals et al. 2015] to generate a caption from the image. The caption generator encoder uses *resnet-152* architecture [He et al. 2016] trained on the ILSVRC competition dataset for the general image classification task [Russakovsky et al. 2015], and the decoder is tuned on COCO-caption [Lin et al. 2014], the same dataset from which we extract all visual context information.

3.4 Dataset Construction

This section describes the construction of the dataset used in this work in two stages: First, we present a simple way to extract and select each text hypothesis from each text image. Second, we use different measures to obtain the visual context information and its co-occurrence in each image. The datasets we created are summarized in Table 3.3.

3.4.1 Text Hypothesis Selection

As described in Section 3.3.1, the output of several text spotting systems is included in the dataset as text hypotheses or possible candidates for each image. However, some filtering is applied to remove duplicates and words that are unlikely to be correct:

First, we use a unigram language model (ULM) to filter out rare words, non-words, or very short words unlikely to be in the image. The ULM was trained

on Opensubtitles [Lison and Tiedemann 2016][2], a large database of subtitles for movies containing around 3 million unique word forms, including numbers and other alphanumeric combinations that make it well suited for our task. Also, we combined this corpus with google-ngram[3] which contains 5 million unique word forms from English literature books. The combined corpora contain around 7 million unique word forms.

Secondly, we add the *ground-truth* text if it was removed by the filter or it was not included in the hypothesis list generated by the baselines. Note that this may occur often, since according to the author of the COCO-text [Veit et al. 2016] the most significant shortcoming of this dataset is bounding box detection recall. Therefore, in about 40% of the images, the text is not properly detected, and thus the classifiers fail to recognize it.

3.4.2 Visual Context Selection

Despite the fact that we extract the top-5 objects from each image, we use a semantic similarity measure and threshold to filter out predictions when the object classifier is not confident enough. We use two approaches to filtering out duplicated cases and false-positive example.

Threshold measure. First, we consider a threshold to extract the most likely classes in the images and eliminate low confidence predictions.

Semantic alignment. We use cosine similarity to select the most likely visual context in the image. In detail, we use general text word-embedding [Mikolov et al. 2013b, Pennington et al. 2014] to compute the similarity score between the different visual context elements. Then we select objects or places that 1) are detected with high confidence, and 2) have a strong semantic relatedness with other image elements. The underlying assumption is that if two objects in the image are related, the object classifier prediction we are relying on will be more likely to be correct.

3.4.3 Object and Text Co-occurrence Database

Finally, we enrich the dataset with co-occurrence information between text and objects. This co-occurrence information should be useful when the text hypotheses and the scenes are not related in the semantic space, but they are in the real world

[2]https://www.opensubtitles.org
[3]https://books.google.com/ngrams

Table 3.5: Data statistic for training dataset that publicly available for caption and text spotting.

Unique Count for Textual Dataset								
Dataset	image #	bbox	caption	object	words	nouns	verb	adjectives
Conceptual [Sharma et al. 2018]	3m	-	3m	-	34,219,055	10,254,864	1,043,385	3,263,654
MSCOCO [Lin et al. 2014]	82k	-	413k	-	3,732,339	3,401,489	250,761	424,977
Flickr 30k [Young et al. 2014]	30k	-	160k	-	2,604,646	509,459	139,128	169,158
SVT [Wang and Belongie 2010]	350	✓	-	-	10,437	3,856	46	666
COCO-text [Veit et al. 2016]	66k	✓	-	-	177,547	134,970	770	11,393
COCO-text-V (ours)	16k	✓	60k	120k	697,335	246,013	35,807	40,922
IC17-V-T3 (ours)	10k	✓	25k	50k	296,500	96,371	15,820	15,023
COCO-Pairs (ours)	66k	-	-	158k	319,178	188,295	6,878	46,983

(for instance *delta* and *airliner* may not be close according to a general word embedding model, but they often co-occur in the image dataset). Figure 3.4 (b) shows some frequency counts of co-occurring text-object pairs.

This co-occurrence information can be used to estimate the conditional probability $P(w|c)$ of a word w given that object c appears in the image:

$$P(w|c) = \frac{freq(w,c)}{freq(c)} \tag{3.5}$$

where $freq(w,c)$ is the number of training images where w appears as the gold standard (gold-truth) annotation for the recognized text, and the object classifier detects object label c in the image. Similarly, $freq(c)$ is the number of training images where the object classifier detects object class c.

3.4.4 Resulting Datasets

In this section, we outline in more detail our textual visual context dataset extension to the COCO-text as shown in Table 3.5.

3.4.4.1 COCO-text with Visual Context

We propose three different visual context datasets for the COCO-text as shown in Table 3.3: 1) training dataset (COCO-text-V), 2) benchmark testing (IC17-V), and 3) Object and text co-occurrence database (COCO-Pairs) as follows:

COCO-text-V: Consists of 16k images with associated bounding boxes, and 60k textual data such as captions, objects and scene visual information for training. As shown in Table 3.4, for each bounding box, we extract k=10 text hypotheses, each of which have different or the same visual context information depending on the semantic alignment (*e.g.* similarity distance). Figure 3.5 show some examples extracted from COCO-text with the visual context information.

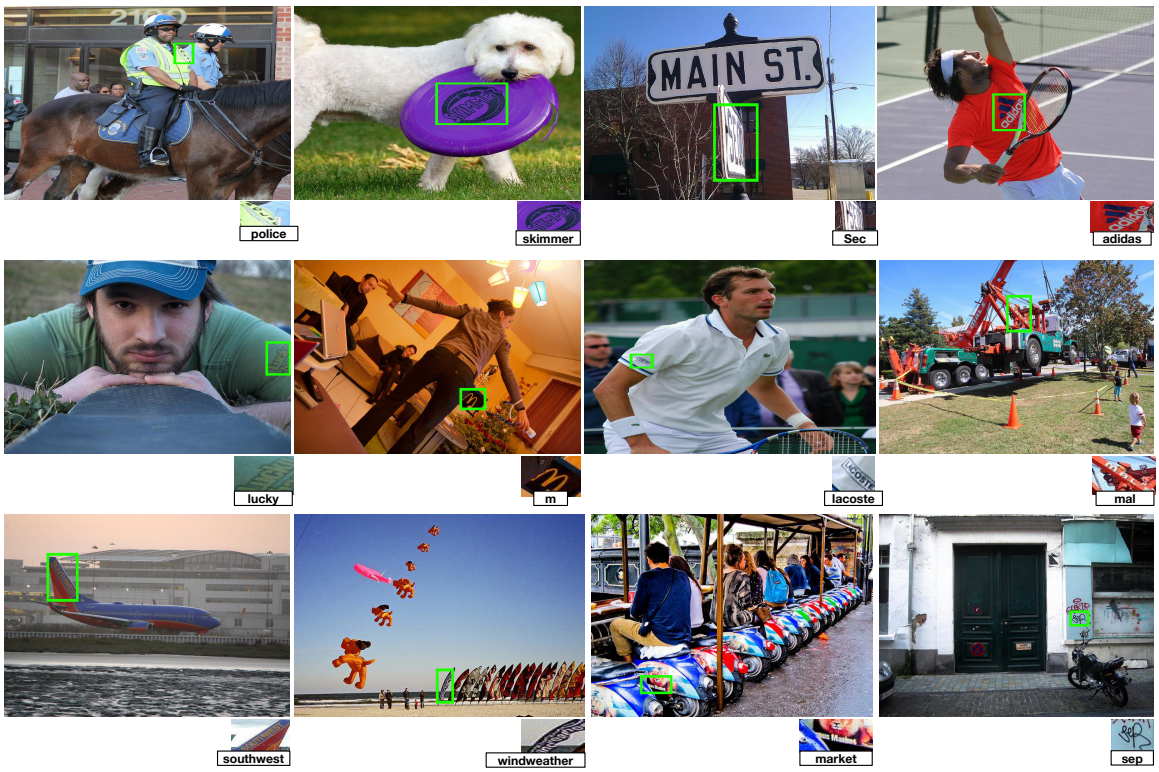

Figure 3.6: Some random examples extracted from COCO-text with poor detection. Poor detection affects the accuracy of our baseline. Thus, we use **ground truth annotation** to overcome this shortcoming in this dataset COCO-text.

ICDAR17-Task3-V (IC17-V-T3) is based on the ICDAR17 Task 3 end-to-end text recognition dataset. Similar to COCO-text-V, we only introduce the visual context (textual dataset) for each bounding box. It consists of 10k images with 25k of textual data for testing and validation.

COCO-Pairs: This textual dataset has no bounding box, only the textual information. The dataset consists of only one pair of object-text extracted from each image. It consists of 158k word-visual context pairs. We combined the output from the visual classifier with the ground truth to create the pairs (*e.g.* text-scene, text-object).

As seen in Figure 3.6 real text in the wild is a very challenging problem and thus, current state-of-the-art approaches, including our dataset struggle to detect the correct coordination of the bounding box. Therefore, we use the **ground truth annotation** to overcome this shortcoming in this dataset COCO-text.

3.5 Task

To evaluate the utility of the proposed dataset, we define a novel task, consisting of using the visual context in the image where the text appears to re-rank a list of candidates for the spotted text generated by some pre-existing model.

More specifically, the task is to use different *similarity* or *relatedness* scorers to reorder the k-best hypothesis produced by a trained model with a softmax output. This candidate word re-ranking should filter out false positive and eliminate low frequency short words. The softmax score and the probabilities of the most related elements in the visual context are then combined by simple algebraic multiplication. In this thesis, we experimented extracting and re-ranking k-best hypotheses for $k = 1 \dots 10$.

3.5.1 Human Evaluation

To calibrate the difficulty of the task we picked 33 random pictures from the test dataset and had 16 human subjects try to select the correct word among the top $k = 5$ candidates produced by the baseline text spotting system. We observed that human subjects more familiar with ads and commercial logos obtain higher scores. Average human performance was 63% (highest 87%, lowest 39%). Figure 3.7 shows the user interface for human annotation.

3.5.2 Evaluation Remarks

For evaluation, we use a less restrictive protocol than the standard one proposed by [Wang and Belongie 2010] and adopted in most state-of-the-art benchmarks, which does not consider words with less than three characters. This protocol was introduced to overcome the false positives on short words that most current state-of-the-art struggle with, including our baseline. Instead, we consider all cases in the dataset, and words with less than three characters are also included.

Since our task is re-ranking, we use the Mean Reciprocal Rank (MRR) to evaluate the quality of re-ranker outputs. MRR is computed as:

$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{k=1}^{|Q|} \frac{1}{\mathrm{rank}_k} \tag{3.6}$$

where rank_k is the position of the correct answer in the re-ranked hypothesis list for image k, Q is the number of images in the dataset. MRR is only looking at the

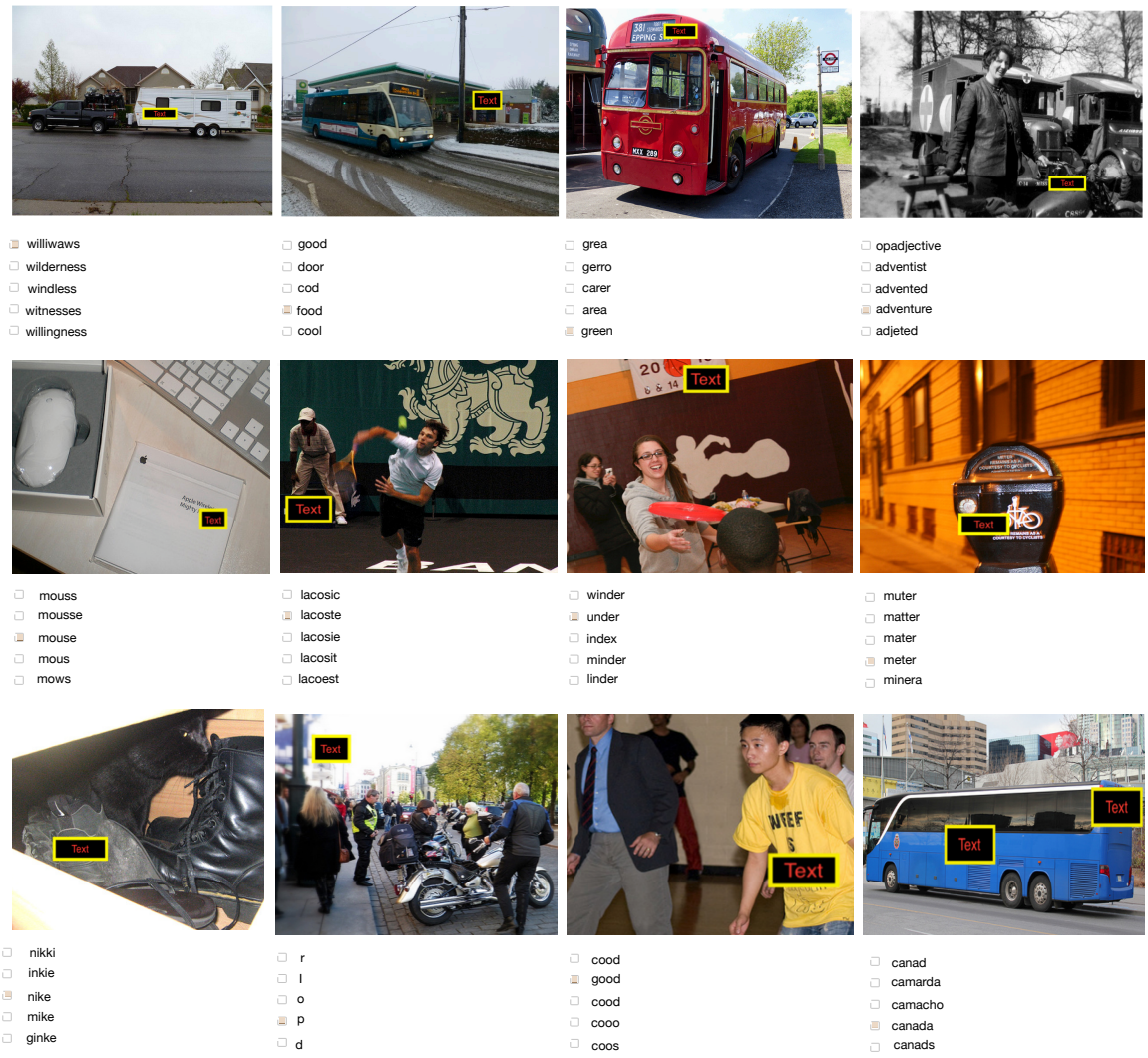

Figure 3.7: The user interface presented to our human subjects through the survey website asking them to re-rank the text hypothesis based on the visual information.

rank of the first correct answer; hence it is more suitable in cases such ours, where for each candidate word there is only a single right answer.

3.6 Conclusion

In this chapter, we have introduced a visual context dataset for the text spotting problem. Unlike other methods that devise a complex architecture to extract visual information, we employ out-of-the-box state-of-the-art tools. Therefore, dataset annotation will be improved in the future as better systems become available. This dataset can be used to leverage semantic relations between image context and can-

didate texts into text spotting systems, either as post-processing, tuning or end-to-end training.

Our dataset would benefit from better bounding box annotation in COCO-text (ideally manual annotation). It also would be useful to extend the provided visual context (currently two objects, a scene/location, and a caption) to include more objects and scene labels, or even captions generated by different systems.

Chapter 4

Learning to Re-rank Text Spotting with Word Embeddings

Most recent state-of-the-art text spotting systems focus on automatically detecting and recognizing text in natural images from a pure computer vision perspective. However, in this chapter we aim to show that leveraging natural language processing techniques such as post-processing can improve the result of the original model without re-training or tuning. Our approach seeks to integrate prior information into the text spotting pipeline.

This idea has been explored recently in a relatively reduced number of works, such as [Kang et al. 2017, Prasad and Wai Kin Kong 2018] which are described in Chapter 2. In this chapter, we will show that by introducing a candidate re-ranker based on word frequencies and semantic distance between candidate words and objects in the image, the performance of an off-the-shelf deep neural network can be improved without the need to perform additional training or tuning. In addition, thanks to the inclusion of the unigram probabilities, we overcome the most recent state-of-the-art limitation of the false detection of short words.

The key novelty of this chapter is that we include several post-processing methods based on NLP techniques such as word frequencies and semantic relatedness, which are typically exploited in NLP problems but less common in computer vision tasks. Another novelty is the use of word embedding as a re-ranker and the proposal of two different methods for obtaining a probability from a similarity score based on the visual context.

4.1 Overview of the Approach

Text recognition approaches can be divided into two categories: (a) character-based methods that rely on a single character classifier plus sequence modeling (*e.g.* n-gram models or LSTMs), and (b) lexicon-based approaches that attempt to classify the image as a whole word.

In both cases, the system can be configured to predict the k most likely words given the input image. Our approach focuses on re-ranking that list using language information such as word frequencies and the semantic relatedness with objects in the image (or *visual context*) in which the text is located, as shown in Figure 4.1.

4.1.1 Baseline Systems

We used two different off-the-shelf baseline models, which are described in Chapter 3. Firstly, we describe a CNN [Jaderberg et al. 2016] with fixed-lexicon based recognition. Here, the lexicon acts as a language prior (dictionary) for the classification task, one class per word. The fixed dictionary contains around 90k word forms (*e.g.* verb, noun, adjective, etc.).

Secondly, we considered an LSTM architecture with a visual attention model [Ghosh et al. 2017]. The LSTM generates the final output words as character sequences, without relying on a lexicon. In particular, the model employs a beam search over the LSTM output to preform word inference.

Both models are trained on a synthetic dataset [Jaderberg et al. 2014b]. The output of both models is a vector of softmax probabilities for candidate words. For each text image, the baselines provide a series of k text hypotheses, that are fed into our model. Let us denote the baseline probability of each of the k most likely words ($w_j, 1 \leq j \leq k$) produced by the baseline as follows:

$$P_{BL}(w_j) \;=\; softmax(w_j, BL) \tag{4.1}$$

4.2 Language Model

Word n-grams capture the conditional probability of a word occurring in a sequence given the previous $n-1$. Estimating the likelihood of the next word is closely related to computing a sequence of words. N-grams are essential in many

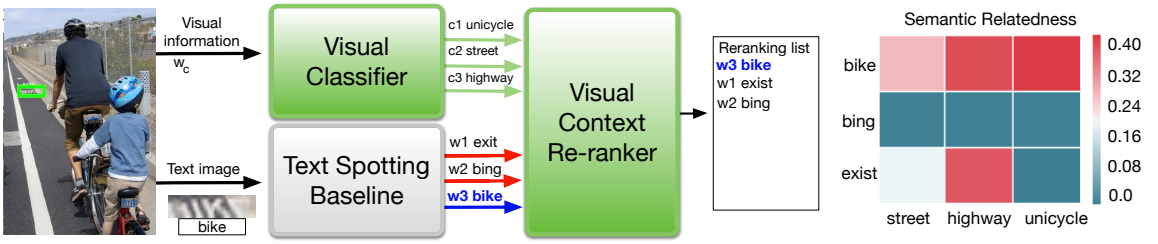

Figure 4.1: Our proposed post-process re-ranks potential words based on their semantic relatedness with the context in the image (objects, scenarios, ...). In the example, the word *bike* has been re-ranked, overcoming other candidates thanks to the detected objects/places (c_i).

tasks that require predicting words in noisy or ambiguous input. For instance, in speech recognition, the input speech is noisy, and most of the words are extremely similar. Shilman and Viola (2004) give an intuition of how the probability of word sequence can help in the case of handwriting recognition. In spelling correction, it may help to find the correct spelling error, as in [Kukich 1992], where techniques are introduced to automatically correct words in a text. N-grams also have applications in part-of-speech tagging, natural language generation, word similarity, and in-text prediction systems such as smart-phone keyboards. In this work, we adopted one of many applications of n-grams, such as spelling correction as a unigram based dictionary.

The probabilities of the Unigram Language Model (ULM) are computed from the *Opensubtitles*[1] corpus [Lison and Tiedemann 2016] and *Google book n-gram*[2] text corpora (7 Million tokens). The main goal of ULM is to increase the probability of the most common words proposed by the baseline.

$$P_{ULM}(w_j) = \frac{count(w_j)}{\sum_{\tilde{w}} count(\tilde{w})} \tag{4.2}$$

It is worth mentioning that the language model is very simple to build, train, and adapt to new domains, which opens up the possibility of improving baseline performance for specific applications. Table 4.1 shows a comparison between our proposed lexicon and the 90k dictionary [Jaderberg et al. 2016].

[1]https://opensubtitles.org
[2]https://books.google.com/ngrams

Table 4.1: Total count of unique words – Lexicon.

Dictionary	words	nouns	verb	adjectives
Dic-90k [Jaderberg et al. 2016]	87,629	20,146	6,956	15,534
ULM (This chapter)	8870,209	2695,906	139,385	824,581

4.3 Visual Context Information

Our second prior for re-ranking the baseline output is based on the visual context information about the image in which the text is located. For this, we use multiple visual classifiers as described in Chapter 3, and devise a strategy to reward the candidate words that are more semantically related to the objects/scenes detected in the image.

4.3.1 Semantic Relatedness Using Word Embedding

Word embeddings have become one of the popular representations of vocabulary semantics. They model word context in a semantic space, in such a way that words with similar contexts are close to each other in the semantic space. In order to make our approach universal, we based our work on general word embedding, such as word2vec [Mikolov et al. 2013b], glove [Pennington et al. 2014] and fasttext [Joulin et al. 2016].

We compute word-embedding based semantic relatedness (SWE) in the following steps: first, we use a threshold δ to eliminate lower probabilities from the visual classifier (objects, scenes); secondly, following Lee et al. (2018) who show the benefits of finding the proper visual context of each region in a caption guided by text using semantic similarity, we compute the embedding-based similarity of each visual with the candidate word. The idea is to match the most semantically related visual context with the candidate word; thirdly, we take the max-highest similarity score and the most semantically related, to the candidate word c_{max} as:

$$c_{max} = \underset{\substack{c_i \in Image \\ P(c_i) \geq \delta}}{\operatorname{argmax}} \; sim(w, c_i) \tag{4.3}$$

where $sim(w, c_i)$ is the cosine similarity, w is the candidate word, c_i is the set of visual object found in the target image, δ is the threshold to eliminate object and scene with lower probabilities.

Table 4.2: Example of visual context based hypotheses revision [Blok et al. 2003]. Marks • and ⋆ show the hypothesis probability before and after the revision, respectively. The heatmap indicates the ratio of changes from the original score.

Word	context	$P(w)$•	P(c)	sim(w,c)	$P(w\|c)$ ⋆
plate	moving	0.0029	0.53	0.30	0.012
electric	streetcar	0.0002	0.48	0.24	0.001
computer	street	0.0023	0.46	0.19	0.007
way	downtown	0.0129	0.35	0.18	0.094
12	football	0.0021	0.68	0.17	0.041
bike	highway	0.0005	0.40	0.44	0.014
walk	street	0.0016	0.30	0.53	0.061
private	bus	0.012	0.45	0.30	0.045

Finally, following [Blok et al. 2003] with confirmation assumption $p(w|c) \geqslant p(w)$, we compute the conditional probability from similarity as introduced in Section 2.3.4:

$$P_{SWE}(w|c_{max}) = P(w)^{\alpha} \tag{4.4}$$

where:

- $\alpha = \left(\frac{1-sim(w,c_{max})}{1+sim(w,c_{max})}\right)^{1-P(c_{max})}$
- $P(w)$ is the probability of the word in general language (*i.e.* a unigram model) computed from *Opensubtitles* [Lison and Tiedemann 2016] and *Google book n-gram*[3] text corpora (7 million tokens).
- $P(c_{max})$ is the probability (as produced by the visual classifier) of the context object/scenario most semantically related to the candidate word.

According to [Blok et al. 2003], the similarity to probability conversion can be defined in terms of **belief revision**, where $P(w)$ is the *hypothesis* probability (the candidate word in our case) and the visual context $P(c)$ is the evidence that causes the hypothesis probability (or belief) to be revised. There are two factors to determine the hypothesis probability revision: 1) the sufficient relatedness to the category – as $sim(w, c)$ goes to 0, α goes to 1, and thus $P(w|c) = P(w)$, *i.e.* no revision takes place (no changes in the original belief), while as $sim(w, c)$ goes to 1, α goes to 0, and the hypothesis probability $P(w)$ is revised and raised closer to 1. And 2) the informativeness of the new information $1 - P(c)$ – as $P(c)$ approaches 1 and in consequence is less informative, α also goes to 1, since there is no new information, and thus no revision is needed either.

[3] https://books.google.com/ngrams

As shown above (Equation 4.4), we use the unigram language model to initialize the hypothesis probability $P(w)$ based on a common observation (general text large corpus). Therefore, the new information from the visual context revises the base hypothesis. Table 4.2 shows examples of hypothesis probability revision based on the visual context. For instance, the probability of word *bike* is raised by the presence of the location *highway* in the image.

4.3.2 Semantic Relatedness Using Dual Word Embedding

The SWE reranker described in Section 4.3.1 uses context information separately from the object or scene classifiers. The reranker described in this section aims to combine both kinds of information in a single step. Also, following the same confirmation assumption $p(w|c) \geqslant p(w)$, we convert the obtained similarity into a conditional probability as follows:

$$P_{DSWE}(w|c_{max_1}, c_{max_2}) = \beta M + (1-\beta)S,$$

where:

$$\begin{aligned}
\beta &= \max \left\{ \begin{array}{l} \text{sim}(c_{max_1}, c_{max_2}) \\ \text{sim}(w, c_{max_1}) \\ \text{sim}(w, c_{max_2}) \\ 1.0 - \text{sim}(w, c_{max_1}) \\ 1.0 - \text{sim}(w, c_{max_2}) \\ P_{SWE}(w|c_{max_1}) \\ P_{SWE}(w|c_{max_2}) \end{array} \right\} \\
M &= max(P_{SWE}(w|c_{max_1}), P_{SWE}(w|c_{max_2})) \\
S &= P_{SWE}(w|c_{max_1}) + P_{SWE}(w|c_{max_2}) \\
&\quad - P_{SWE}(w|c_{max_1}) \times P_{SWE}(w|c_{max_2})
\end{aligned} \tag{4.5}$$

where P_{SWE} is defined by Equation 4.4, c_{max_1} and c_{max_2} are the object and place label class, respectively (from the visual classifier). Note that Equations 4.4 and 4.5 already include frequency information from the ULM; therefore they take into account not only the semantic relatedness but also the word frequency information used in the ULM re-ranker (see Section 4.2). Also, if there is no available visual context information, we back-off to $\alpha = 1$ and use the bare unigram probability. β takes the max of all models, and thus it is not breaking the formation if one of the similarity or probability is not confident enough (*i.e.* if it is below the threshold).

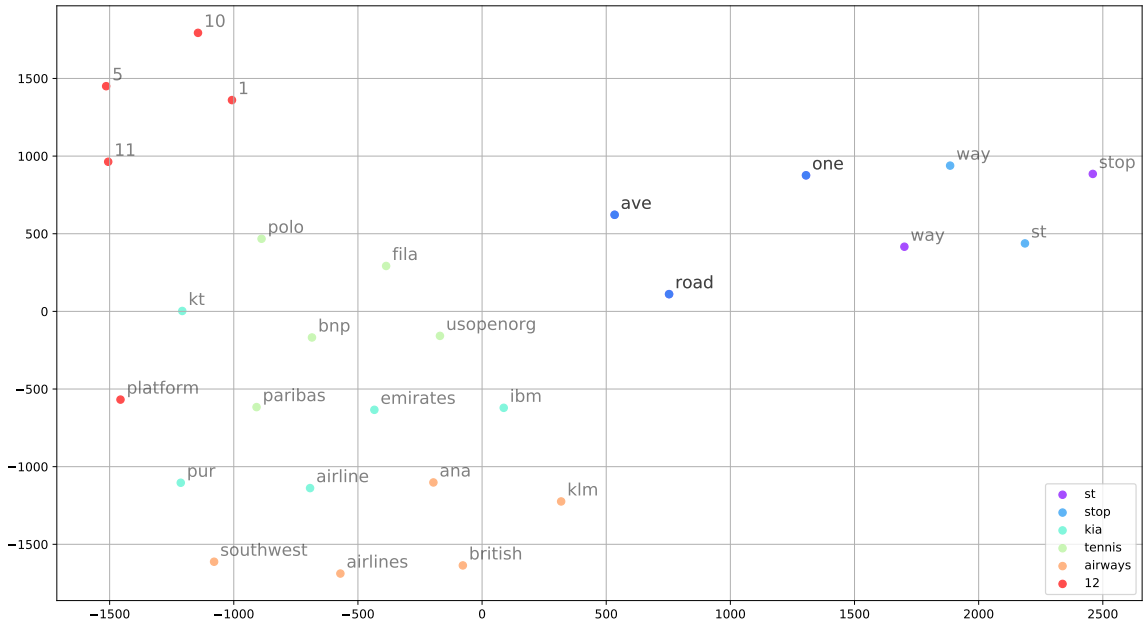

Figure 4.2: Subset of top-k nearest neighbor sample word vector space of TWE model reduced to two dimensions using t-SNE [Maaten and Hinton 2008]. **Boldface** fonts show an example of the top-3 nearest neighbor words that appear, in the street, near the spotted text *stop* sign. Another example, the spotted text *kia* is close to *tennis* and tennis sponsors such as *emirates*, *polo* and sport channel *kt*. The word *way* is near other words that appear in the street such as *one* way, *st* and *stop*. Numbers also appear near words that have a numbering system such as *platform*. The word *airways* is also near to other airline companies, such as *klm* and *ana*.

4.3.3 Training-Data Word Embeddings

This re-ranker builds upon a word embedding, like the SWE re-ranker above, but the embeddings are learned from the training dataset (considering two-word "sentences" consisting of the target word and the object/scene in the image). The embeddings can be computed from scratch, using only the training dataset information (TWE). At inference time, we only consider one visual context at a time similar to SWE.

In particular, we train a skip-gram model [Mikolov et al. 2013b] with negative sampling loss to predict the *context* w_c word, given a center word w. In our case, the context word is the visual information, while the given word is the text hypothesis. Figure 4.2 shows the top-k nearest neighbor words in the vector space. For instance, the word *way* is closer to *st* and *stop* which occur in the street more often.

Since the TWE is confident about the similarity score, we convert the similarity

produced by the embeddings to probabilities using:

$$P_{TWE}(w|c) = \frac{\tanh(sim(w,c)) + 1}{2P(c)} \tag{4.6}$$

where $sim(w,c)$ is the cosine similarity. Note that since $\tanh(x) \in [-1,1]$, then $\tanh(x)+1 \in [0,2]$, and thus $\frac{\tanh(x)+1}{2} \in [0,1]$ is our approximation of $P(w,c)$. This is then divided by $P(c)$ to obtain the conditional probability.

Note that this re-ranker does not take into account word frequency information as in the case of the SWE re-ranker.

4.3.4 Estimating Relatedness from Training Data Probabilities

A second way of computing semantic relatedness is to estimate it from training data (TDP). This should overcome the word embedding limitation when the candidate word and the image objects are not semantically related in general text, but are in the real world. For instance, as shown in Figure 4.4, the sports TV channel *kt* and the object *racket* have no semantic relation according to the word embedding model SWE, but they are found paired multiple times in the training dataset, which implies that they do have a relation. The TDP is computed as follows:

$$P_{TDP}(w|c) = \frac{count(w,c)}{count(c)} \tag{4.7}$$

where $count(w,c)$ is the number of training images where w appears as the gold standard annotation for recognized text, and the object classifier detects object c in the image. Similarly, $count(c)$ is the number of training images where the object classifier detects the object c.

4.4 Combining Expert Re-ranker

Product of Experts (PoE) [Hinton 1999] is an effort to combine the expertise of each expert (model) in a collaborative manner. It allows each expert to specialize in analyzing one particular aspect of the problem and to establish a judgment based on that aspect. In our case, we have three main experts: 1) the Language Model for filtering out false positive words, 2) Word Embedding (*i.e.* SWE/DSWE and TWE) to learn the semantic relatedness and 3) TDP to overcome some of the word embedding out-of-vocabulary (a.k.a. OOV) word limitations and learn the semantic relation that happens in the real world. Therefore, inspired by PoE, we

combine different re-rankers (or *experts*). PoE takes advantage of each expert and can produce much sharper distributions than a single model:

$$P\left(\mathbf{w}|\theta_1 \ldots \theta_n\right) = \frac{\Pi_m p_m\left(\mathbf{w}|\theta_m\right)}{\sum_{\mathbf{c}} \Pi_m p_m\left(\mathbf{c}|\theta_m\right)} \tag{4.8}$$

where θ_m are the parameters of each model m, $p_m(\mathbf{w}|\theta_m)$ is the probability of $\mathbf{w}$ under model m and $\mathbf{c}$ is the indexes of all possible vector in the data space.

Since we are just interested in retrieving the candidate word with higher probability after re-ranking, we do not need to normalize. Therefore, we compute:

$$\arg\max_{\mathbf{w}} P\left(\mathbf{w}|\theta_1 \ldots \theta_n\right) = \arg\max_{\mathbf{w}} \frac{\Pi_m p_m\left(\mathbf{w}|\theta_m\right)}{\sum_{\mathbf{c}} \Pi_m p_m\left(\mathbf{c}|\theta_m\right)} \tag{4.9}$$

$$= \arg\max_{\mathbf{w}} \Pi_m p_m\left(\mathbf{w}|\theta_m\right) \tag{4.10}$$

where, in our case, $p_m\left(\mathbf{w}|\theta_m\right)$ are the probabilities assigned by each expert to the candidate word $\mathbf{w}$.

Our re-ranking approach consists of taking the softmax probabilities computed by the baseline DNN and combining them with the probabilities produced by the re-ranker methods described in Section 4.3. We combine them with simple multiplication, which allows us to combine any number of re-rankers in cascade. For simplicity, in the equations below C stands for the set of available context information (*i.e.* object, scenario). We evaluated the following combinations:

1. The baseline output is re-ranked by the unigram language model.

$$P_0(w) = P_{BL}(w) \times P_{ULM}(w) \tag{4.11}$$

2. The baseline output is re-ranked by the relatedness estimated from the training dataset as conditional probabilities (TDP).

$$P_1(w, c) = P_{BL}(w) \times P_{TDP}(w|C) \tag{4.12}$$

3. The baseline output is re-ranked by the general word-embedding model (SWE/DSWE). Note that these rerankers also includes the ULM information as described in Section 4.3.1.

$$P_{2a}(w, c) = P_{BL}(w) \times P_{SWE}(w|C) \tag{4.13}$$

$$P_{2b}(w, c) = P_{BL}(w) \times P_{DSWE}(w|C) \tag{4.14}$$

4. The baseline output is re-ranked by the word-embedding model trained entirely on training data (TWE).

$$P_3(w,c) = P_{BL}(w) \times P_{TWE}(w|C) \tag{4.15}$$

5. The baseline output is re-ranked by SWE/DSWE general word embedding and TDP re-rankers combined.

$$P_{4a}(w,c) = P_{BL}(w) \times P_{SWE}(w|C) \times P_{TDP}(w|C) \tag{4.16}$$

$$P_{4b}(w,c) = P_{BL}(w) \times P_{DSWE}(w|C) \times P_{TDP}(w|C) \tag{4.17}$$

6. The combination of TDP and TWE (with or without ULM).

$$P_{5a}(w,c) = P_{BL}(w) \times P_{TDP}(w|C) \times P_{TWE}(w|C) \tag{4.18}$$

$$P_{5b}(w,c) = P_{BL}(w) \times P_{ULM}(w) \times P_{TDP}(w|C) \times P_{TWE}(w|C) \tag{4.19}$$

4.5 Experiments and Results

This section evaluates our method using standard text spotting benchmarks. Data and preliminaries are discussed next, experiments are discussed in Section 4.5.4, and we compare our model with state-of-the-art models in Section 4.5.5. A full description of the datasets referenced can be found in Chapter 3.

4.5.1 Dataset

We evaluate our pipeline on a COCO based dataset [Lin et al. 2014] for training the COCO-text [Veit et al. 2016] and for evaluating ICDAR17 [Gomez et al. 2017], which are described in Chapter 3.3. In particular, we train our model on a combined corpus of COCO-text-Visual and COCO-Pairs: 1) COCO-text-V, consisting of 16k images with associated bounding boxes, and a 60k textual dataset including 120k objects and scenes class labels, and 2) COCO-Pairs, which has no bounding box, only the textual information. The dataset consists of only a pair of object-texts extracted from each image. It consists of 158k word-visual context pairs. We evaluate our model on ICDAR17-Task3-V, a dataset that is based on the ICDAR17 Task 3 end-to-end text recognition.

4.5.2 Preliminaries

For evaluation, we used a more restrictive protocol than the standard proposed by [Wang et al. 2011] and adopted in most state-of-the-art benchmarks, which does not consider words with fewer than three characters or with non-alphanumeric characters. This protocol was introduced to overcome the issue with false positives on short words that most current state-of-the-art struggle with, including our baseline. However, we overcame this limitation by introducing the language model re-ranker. Thus, we consider all cases in the dataset, and words with fewer than three characters are also evaluated.

In all cases, we use two pre-trained deep models, CNN [Jaderberg et al. 2016] and LSTM [Ghosh et al. 2017] as a baseline (BL) to extract the initial list of word hypotheses. Since these BLs need to be fed with the cropped words, when evaluating on the ICDAR-2017-Task3 dataset, we use the ground truth bounding boxes of the words.

4.5.3 Experiment with Language Model

As a proof of concept, we built our unigram language model on two different corpora. The first ULM was trained on *Opensubtitles*[4], a large database of subtitles for movies containing around three million word types. Opensubtitle corps contain a variety of different text such as numbers, colloquialism, non-standard words and alphanumeric characters, making it well suited for our task. Secondly, we built another model with *Google book n-gram*[5], which contains five million word types from American-British literature books. We used this corpus to increase the frequency of the most common words that are not seen in the other corpus. We use a combined version of both corpora, which contains around seven million types of words (token).

In this experiment, we extract the $k = 1, \ldots, 10$ most likely words – and their probabilities – from the baselines. Although the sequential nature of the LSTM baseline captures a character-based language model, our post-process uses word-level probabilities to re-rank the word as a whole (most current state-of-the-art use word-level approach). Note that since our baselines work on cropped words, we do not evaluate the whole end-to-end but only the influence of adding external knowledge.

[4] *https://www.opensubtitles.org*
[5] *https://books.google.com/ngrams*

Table 4.3: Best results after re-ranking using different re-ranker combinations, and different values for k-best hypotheses extracted from the baseline output.

(Re-ranker) Model	**CNN**				**LSTM**		
	full	*dict*	*list*	*k*	*full*	*list*	*k*
Baseline (BL)	**full: 19.7 dict: 56.0**				**full: 17.9**		
(P_0) BL+LM_{7M}	21.9	62.4	75.2	7	19.3	76.2	4
(P_1) BL+$TDP_{objects}$	20.4	58.0	77.8	4	18.8	74.0	4
(P_{2a}) BL+SWE_{object}	21.9	62.4	**86.2**	4	19.1	75.3	4
(P_{2a}) BL+SWE_{place}	22.0	62.5	75.8	7	18.7	73.6	4
(P_{4a}) BL+$TDP_{objects}$+SWE_{object}	22.2	63.2	81.4	5	19.5	76.9	4
(P_{4a}) BL+$TDP_{objects}$+SWE_{place}	22.3	63.5	76.9	7	19.8	66.4	9
(P_{2b}) BL+$DSWE_{objects}$	21.9	62.4	78.1	6	19.0	74.9	4
(P_{2b}) BL+$DSWE_{places}$	21.9	62.2	75.4	7	18.4	72.5	4
(P_{2b}) BL+$DSWE_{object+place}$	21.9	62.2	75.4	7	19.3	76.0	4
(P_{4b}) BL+$TDP_{objects}$+$DSWE_{objects}$	22.2	63.0	84.6	4	19.5	76.9	4
(P_{4b}) BL+$TDP_{objects}$+$DSWE_{places}$	22.2	63.0	76.3	7	19.1	75.3	4
(P_{4b}) BL+$TDP_{objects}$+$DSWE_{object+place}$	22.2	63.0	81.2	5	19.6	77.3	4
(P_3) BL+TWE_{object}	22.2	76.3	63.0	7	19.5	76.9	4
(P_{5a}) BL+TDP_{object}+TWE_{object}	**22.5**	63.8	77.3	7	**19.7**	**77.5**	4
(P_{5b}) BL+TDP_{object}+TWE_{object}+LM_{7M}	22.1	62.9	84.4	4	19.3	76.0	4

The first baseline is a CNN [Jaderberg et al. 2016] with fixed-lexicon recognition, which is not able to recognize any word outside its dictionary. The results are reported in Table 4.3. We present three different accuracy metrics: *full* columns correspond to the accuracy achieved on the whole dataset, while the *dictionary* columns correspond to the accuracy achieved in the solvable cases (*i.e.* those where the target word is among the 90k-words in the CNN dictionary, which correspond to 43.3% of the whole dataset), and the *list* columns, which report the accuracy achieved in the cases where the right word was among the k-best produced by the baseline.

We also provide the results using different numbers of the k-best candidates. In Table 4.3 the top row shows the performance of the CNN baseline, while the second row reports the influence of the ULM. The best result is obtained with $k = 7$, which improved the baseline model by 2.2% *full*, 6.4% *dictionary* and retrieved 75.3% of the correct text hypotheses.

The second baseline we consider is an LSTM [Ghosh et al. 2017] with visual soft-attention mechanism, which performs unconstrained text recognition without relying on a lexicon. The first row in Table 4.3 reports the LSTM baseline result achieved on this dataset, while the second row shows the results after the ULM

w	Text Spotting Model		Visual Re-ranker Model
w_1	quotas	0.5	5.4e-7
w_2	quartos	0.1	5.2e-8
w_3	**quarters**	0.05	9.0e-9

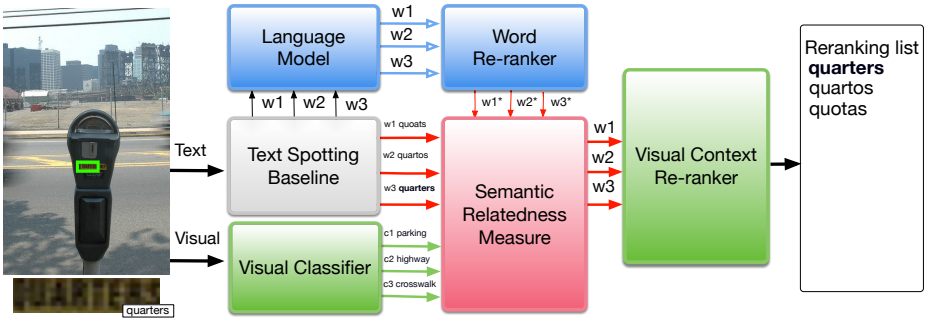

Figure 4.3: Scheme of the proposed visual context information pipeline integration into the text spotting system. Our approach uses the language model and a semantic relatedness measure to re-rank the word hypothesis. The re-ranked word *quarters* is semantically related to the top-ranked visual *parking*. The table above shows the modified probabilities before and after the visual context. As shown, these individual word scores are modified (re-ranked) based on the presence of other object and scene classes which are extracted through state-of-the-art frameworks from each problem domain.

re-ranking. The best results are obtained by considering $k = 4$ which improves the baseline by 1.4% *full* and achieves a 76.2 % retrieval score.

4.5.4 Experiment with Visual Information

The main contribution of this chapter consists of re-ranking the k most likely hypotheses candidate word using the visual context information. Thus, we use the ICDAR-2017-Task3 dataset to evaluate our approach, re-ranking the baseline output using the semantic relation between the spotted text in the image and its visual context. As in the language model experiment, we used ground-truth bounding boxes as input for the BL. However, in this case, the whole image is used as input for the visual classifier, as shown in Figure 4.3.

In order to extract the visual context information we considered two different pre-trained state-of-the-art visual classifiers: object and scene classifiers. For image classification, we rely on three pre-trained networks: ResNet152 [He et al. 2016], GoogLeNet [Szegedy et al. 2015] and Inception-ResNet-v2 [Szegedy et al. 2017], all of which able to detect pre-defined list of 1,000 object classes. However, for testing

Table 4.4: Examples of P(word|object + scene) for each re-ranker. Semantic Relatedness Using Dual Word Embedding (DSWE) captures most relevant information for improving the baseline from both object and scene information. The triangle △ indicates the commercial brand.

Word	Object	Place	SWE_{object}	SWE_{place}	DSWE
way	**street**	**hospital**	0.1771	0.3425	**0.3787**
canada	**passenger**	**bus**	0.0004	0.0001	**0.0005**
member	**ballplayer**	**baseball**	0.0006	0.0009	**0.0011**
bank	**traffic**	**motel**	0.0112	0.1079	**0.1361**
3	**ballplayer**	baseball	**0.0075**	0.002	0.0072
dunkin △	**coffee**	**bakery**	0.0250	0.0142	**0.0309**
w △	racket	**stadium**	9.88e-5	**2.53e-4**	2.51e-4
sony △	**racket**	athletic	**0.0154**	1.09e-4	1.05e-4

we considered only Inception-ResNet-v2 due to its better *top-5 accuracy*. For scene classification we used *Place365-ResNet152* places classifier [Zhou et al. 2017] which is able to detect 365 scene categories.

Although the visual classifiers use a softmax to produce only one probable object hypothesis per image, we use *threshold* to extract a number of object-scene hypotheses and eliminate low-confidence results. Then we compute the semantic relatedness for each object-scene hypothesis with the spotted text.

In this experiment, we re-rank the baseline k-best hypotheses based on their relatedness with the objects/scenes in the image. To do so, we try three approaches : 1) semantic similarity computed using general word embeddings (*i.e.* word2vec, glove, fasttext), 2) correlation based on co-occurrences of text and image object in the training data, and 3) semantic similarity from scratch on the training data.

First, we re-rank the words based on their word embedding: their semantic relatedness with multiple visual contexts from the general text: 1) object (SWE_{object}) and 2) scene (SWE_{place}). For instance, the example in Figure 4.3 and the table above show that the strong semantic similarity between the scene information *quarters* and *parking* re-ranked that word from 3rd to 1st position. We tested three pre-trained models trained on general text as a baseline 1) word2vec model [Mikolov et al. 2013b] with 100 billion tokens 2) glove model with 840 billion tokens [Pennington et al. 2014] and 3) fastText [Joulin et al. 2017] with 600 billion tokens. However, we adopted glove as a baseline due to its better similarity score.

Secondly, we re-rank words based on a combined SWE (objects , places) in a single step (Dual SWE). DSWE uses two visual information (object, scene) to compute

Table 4.5: Examples of P(word|object) for each re-ranker. TDP and TWE capture relevant information for improving the baseline for the pairs of word-object/scene that appear in the training dataset. The TPD overcomes word-embedding limitations in samples occurring in training datasets. The Triangle △ indicates the commercial brand.

word (w)	context (c)	$P_{SWE}(w\|c)$	$P_{TDP}(w\|c)$	$P_{TWE}(w\|c)$
delta △	airliner	0.0028	**0.0398**	0.0003
kt △	racket	0.0004	**0.0187**	0.0002
plate	moving	0.0129	0.0005	**0.326**
way	street	0.1740	0.0216	**0.177**

the relation between the two visual contexts and the candidate words. For example, in Table 4.4 DSWE was able to re-rank the candidate word **dunkin** donuts store, based on the detection of the context *coffee* and *bakery*. Also, we use DSWE with two objects from different classifiers namely Inception-ResNet-v2 [Szegedy et al. 2017] and Resnet152 [He et al. 2016]. In addition, extracting the top-2 k words from the same classifier performs better in the case of $DSWE_{places}$.

In particular, the results in Table 4.3 show that DSWE alone improves the CNN accuracy. However, SWE (single embedding) yields a better performance on both baselines, since DSWE struggles with short words and false positives. For instance, as shown in Table 4.4, the candidate one-letter word **w** (tennis racket brand **W**ilson) has a low semantic relatedness score with the visual contexts *racket* and *stadium*.

Thirdly, we use the training data to compute the conditional probabilities of the text image and the object in the image occurring together (TDP). We also combined both relatedness measures as described in Equation 4.16, obtaining greater improvement in accuracy on both baselines. As can be seen in Table 4.3, (SWE+TDP) boosted the accuracy for both baselines. For example, as shown in Figure 4.4 the *kt* (sport channel) occurs frequently within the visual context *racket*, something that can not be captured by general word embedding models. Also, the scene classifier SWE_{place}+TDP boosts the baseline. The scene classifier SWE_{place} performs better than the object classifier in outdoor setting. For instance, the spotted text *way* in a signboard is more semantically related with *downtown*, *street* or *plaza* than *an umbrella* in the image.

Fourthly, we trained a word embedding model [Mikolov et al. 2013b] using the training dataset (TWE). Since the dataset is very small, we trained a skip-gram model with two windows, and without any word filtering. The result is a 300-

dimension vector for around 200k words. The results in Table 4.3 CNN show that the combination of model TDP+TWE also significantly boosts the accuracy up to 7.8% dictionary, 2.8% *all* and 77.3% retrieval. Also, the second baseline LSTM accuracy boosted up to 1.8 % and 77.5% retrievals. It is also worth mentioning that the TDP+TWE model only relies on the visual context information, as computed by Equation 4.6.

Table 4.5 shows an example of the proposed models stand-alone. Since the TWE was trained on the same dataset, it captures more semantic information. The TPD model overcomes some of the word embedding OOV limitations.

4.5.5 Comparison with State-of-the-Art

Finally, as shown in Table 4.6 we also compare our result with current state-of-the-art word embeddings trained on a large general text using *glove*, *fasttext* and an enhanced version of word-embedding (*i.e.* sense and relational word embeddings).

4.5.5.1 General Word Embedding

We first give the details of the approaches that we will compare ours with:

GloVe [Pennington et al. 2014]: A word embedding system that overcomes the limitation of the original word2vec [Mikolov et al. 2013b]. The main idea of glove is that it can derive the semantic relationships between words from the co-occurrence matrix. The advantage of glove over word2vec is that it does not rely on local information, such as the local context of words, but rather incorporates global co-occurrence statistics. For that reason, glove can derive better semantic relationships than other models. We evaluated the model with the common crawl cased general glove embedding 300-Dimension with 840 billion tokens and 2.2 million vocabularies.

Fasttext [Joulin et al. 2017]: Fasttext is an extension of the word2vec model. Its main characteristic is that instead of learning the word directly, like word2vec, an n-gram character representation is learned. Thus, it can deal with rare words not seen during training by breaking them down into character n-grams to get their embeddings. In short, the model generates better embeddings for unseen words. In particular, after a word has been represented as a character n-grams set, a skip-gram model is trained to learn the word-level embedding. Although Fasttext overcomes some of the limitations of word2vec, such as unseen words, it is unable to generate a word similarity vector representation as good as that of

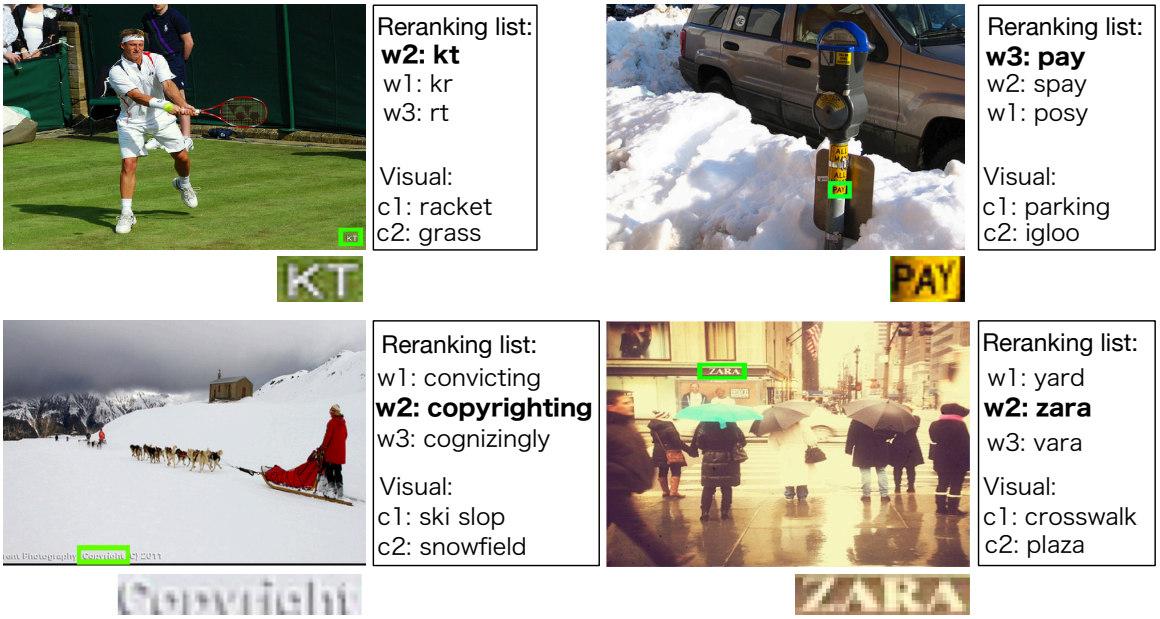

Figure 4.4: Some examples of the visual context re-ranker. The top-two examples are successful results of the visual context re-ranker. The top-left example (*kt*, racket) is a re-ranking result based on the relation between text and its visuals that occur together in the training dataset. The top-right example (*pay*, parking) is a re-ranking result based on the semantic relatedness between the text image and its visual. The bottom two cases (*zara*, crosswalk), (*copyright*, ski) are examples of words that either have no semantic correlation with the visual or that exist in the training dataset. Note that the top ranking visual c_1 is the most semantically related visual context to the spotted text. (**Boldface** font reflect the ground truth).

glove or word2vec. In our case, we use the common crawl 600 billion tokens and two million word vectors trained with subword information.

4.5.5.2 Knowledge-Enhanced Word Embedding

One of the limitations of the word embedding systems that we mentioned above is that they compute a single representation for each word independently of the context in which they appear, context insensitive. In this section, we describe several recent approaches that attempt to enhance word embedding by adding external knowledge.

Senses and Words to Vectors (Sw2v) [Mancini et al. 2017]: Sw2v[6] addresses this issue by proposing a model that learns words with different meanings (sense embeddings) jointly in the same space. In particular, it uses external knowledge from BabelNet [Navigli and Ponzetto 2012] to extract multiple senses for each word.

[6]CBOW architecture of Word2Vec.

Table 4.6: Best results after re-ranking using different state-of-the-art re-rankers, and different values for k-best hypotheses extracted from the baseline output (%).

Model	CNN	k	list	LSTM	k	list
Baseline	19.7	-	-	17.9	-	-
BL+Glove [Pennington et al. 2014]	22.0	7	75.8	19.1	4	75.3
BL+W2v [Mikolov et al. 2013b]	21.8	5	80.0	19.5	4	76.9
BL+Fasttext (Ft) [Joulin et al. 2017]	21.9	7	75.4	19.4	4	76.1
BL+Sw2v [Mancini et al. 2017]	21.8	7	75.2	19.4	4	76.4
BL+LSTMEmbed [Iacobacci 2019]	21.6	7	73.7	19.2	4	75.8
BL+RWE-W2v [Camacho et al. 2019]	21.9	7	75.6	19.6	4	77.3
BL+RWE-Ft [Camacho et al. 2019]	22.0	7	75.8	19.5	4	77.1
BL+TWE (This chapter)	22.2	7	76.3	19.5	4	76.7
BL+TDE+TWE (This chapter)	**22.5**	7	77.3	**19.7**	4	**77.5**

BabelNet is the largest semantic network with a multilingual encyclopedic dictionary, comprising approximately 16 million entries for named entities linked by semantic relations and concepts. In order to employ this model, we obtain senses for the visual context using the same procedure as in Mancini et al. (2017). Each class label has multiple senses that are filtered out using cosine similarity to the closest candidate word. We also introduce senses to image label classification object/place classes, Resent [He et al. 2016] and Places365 [Zhou et al. 2017], which can be used in other applications, such as image or place classification. We employ 300-Dimension word and sense embeddings trained on Wikipedia to compute the similarity distance between each vector (*i.e.* sense, word).

LSTMEmbed [Iacobacci 2019]: LSTMEmbed is the most recent model in sense word embedding, and is similar to Sw2v. It utilizes a BiLSTM architecture to learn the word and sense representations from annotated corpora. The advantage of LSTMEmbed over Sw2v is that it takes word ordering into account during the learning process. However, in our case, word ordering is not essential as we are interested in word-to-context only, and thus LSTMEmbed performs poorly in our evaluation. We use the same approach mentioned above with regard to the Sw2v model to evaluate our model with 200-Dimension senses word embeddings trained on the English language portion of BabelWik and English Wikipedia.

Relational Word Embeddings [Camacho et al. 2019]: (RWE) is an enhanced version of a word embedding model that adds encoded complementary relational knowledge to standard word-embedding in the semantic space. This enhanced embedding is still learned from pure co-occurrance statistics and does not rely on

Table 4.7: Comparison between knowledge base embedding and count-based embedding. v_n^1 indicates the sense embedding.

Word	Visual	$sense_n^1$	sim(w,v)	sim(w,s_n^1)
under	kimono	robe	0.14	0.17
electric	streetcar	vehicle	0.24	0.15
design	bathroom	bathroom	0.33	0.10
station	street	street	0.21	0.23
year	residential	district	0.24	0.09
riding	ski	sport	0.40	0.30

any external knowledge. In particular, the model intends to capture and combine new knowledge that is complementary to that of standard similarity-centric embeddings. We employ the word-embedding and fasttext based version with 300-Dimension trained on English Wikipedia.

Table 4.7 shows that count-base embedding leverage better similarity score in the text spotting scenario. For example, (*ski*, *riding*) has a better semantic similarity score in the general text than the extracted sense *sport* for *ski*.

Also, we experiment with sentence encoders[7], such as Universal Sentence Encoder (USE) [Cer et al. 2018], current state-of-the-art in Semantic Textual Similarity (STS), and Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2019] with fine-tuning and extracting feature to compute the semantic relation with cosine distance. We will discuss in more detail these models in the next chapter (*i.e.* sentence-level encoder).

4.5.6 Discussion

Visual context information re-ranks potential candidate words based on their semantic relatedness with visual information. However, there are some cases when there is no direct semantic correlation between the visual context and the potential word. Thus we proposed TDP (in which semantic relatedness is estimated from training data) to address this limitation by learning correlations from the training dataset. In addition, we overcome the limitation of the word embedding model by adding more information, such as an estimation of how often words occur in that image. However, there are still cases that remain unseen in the training dataset, for instance, as shown in Figure 4.4, the brand name *zara* and the visual context *crosswalk* or *plaza*.

[7]Following the same experimental sitting in a word-level manner.

Limitation. One limitation of the SWE/DSWE approach is that when the text in images is not related to its environmental context, the language model re-ranks it based on general text and word frequencies, which is impractical for rare words or unseen word in the corpus. This is also the case when the false positive in language model is stronger than the visual context re-ranker. For instance, the word *ohh* has a large frequency count in general text. This problem can be tackled by adjusting the weight of uncommon short words in the language model.

4.6 Conclusion

In this chapter, we have presented a simple-post processing approach in the shape of a hypothesis re-ranker based on visual context information with the aim of improving the accuracy of any pre-trained text spotting system. We also show that the performance of the visual context re-ranker can be improved by integrating a hybrid re-ranker, deep learning and statistical language modeling, that based on natural language understanding as a prior to the visual re-ranker. We have shown that the accuracy of two state-of-the-art deep network architectures, a lexicon-based and a lexicon-free recognition architecture, can be boosted up to 2.8 percentage-points on standard benchmarks.

In the next chapter we describe end-to-end based fusion schemes that can automatically discover more proper priors in one single deep fusion architecture.

Chapter 5

Leaning to Re-rank Text Spotting with Neural Network

The approach we described in the previous chapter is based on statistical language modeling and word embeddings. However, the chains of re-rankers used are not trained in an end-to-end fashion and rely on a mixture of different experts such as similarity measures and language models, as shown in Figure 4.3. Also, the proposed approaches suffers from some limitations, such as the influence of the quality of the language model on the word embeddings re-ranker. One of the main drawbacks of word embeddings is their context-insensitively, as can be seen in the following example:

`Rosa attended the` **`play`** `at the firework festival.`

`Rosa did not want to` **`play`** `card-games at the festival.`

In the context-insensitive embedding $f($**`play`**$)$ is the same for each sentence, despite the significant different in meaning. To help alleviate this limitation, in this chapter, we propose an end-to-end neural approach that employs **both** word and sentence semantic relatedness measures as a context-aware embedding model.

As we described in Chapter 2, deep learning has been successful in tasks related to deciding whether two short pieces of text refer to the same topic, such as semantic textual similarity and textual entailment [Parikh et al. 2016], as well as answer ranking for Q&A [Severyn and Moschitti 2015]. However, other tasks require a broader perspective to decide whether two text fragments are *related* more than whether they are *similar*. In this chapter, we describe one such task, and we retrain some of the existing **sentence similarity** approaches to learn this semantic relatedness. In particular, we further exploit additional visual information obtained from

textual descriptions of the image (caption). We also introduce a new deep learning architecture to perform a fusion of all re-rankers.

5.1 Overview of the Approach

In Chapter 4, we described how our semantic-based re-ranker was based on object and scene information extracted from the image. In this chapter, we introduce a caption generator that uses an encoder and a decoder to generate a **synthetic natural language description**. The encoder was trained on the ILSVRC competition dataset for general image classification tasks [Russakovsky et al. 2015], and the decoder is tuned on COCO caption dataset [Lin et al. 2014]. A full description of the caption generator can be found in Chapter 3.

As we described in Chapter 4, given an input image with a text on it, we assume that a baseline approach provides a ranked list of potential words $w_1 \dots w_k$. Our goal is to leverage the context information of the image to re-rank the list, moving the best-k words to the top of the list while removing false-positive candidates. In this chapter we will experiment with computing this relatedness at word-level (using image objects and places) as well as at sentence-level (using image captions), and with a combination of both. We follow the same setup and baselines mentioned in Chapter 4: we will use [Jaderberg et al. 2016, Ghosh et al. 2017] as the baselines that estimate the initial list, but our approach is applicable to any other text-spotting approach that provides a list of candidate words. A schematic of the system is shown in Figure 5.1.

5.2 Architecture

This section discusses the details of learning various semantic based word re-rankers in an end-to-end fashion. To learn the semantic relatedness between the visual context information and a given candidate word, we consider an architecture consisting of a multi-channel convolutional LSTM with an attention mechanism, as shown in Figure 5.2.

The network is fed one candidate word at a time plus several words describing the image visual context (object/place labels and a descriptive caption) and is trained to provide a relatedness score for the candidate word and the context. All visual context information is automatically generated using off-the-shelf existing modules (as described in Chapter 3).

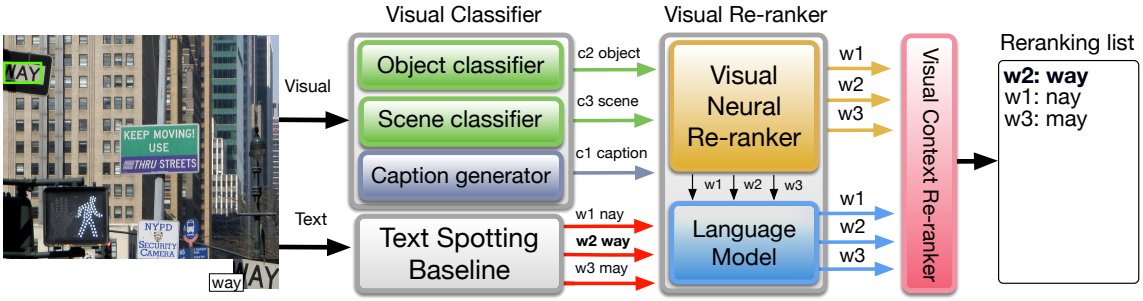

Figure 5.1: System Overview. We propose a neural-based simple post-process methodology that can be applied to the output of any pre-trained state-of-the-art DNN. Our system re-ranks the candidate words using their semantic relatedness with contextual image information such as objects, scene, and textual captions. In the example, the word *way* has been correctly re-ranked after exploring the visual information of the image (like downtown or street).

Our architecture is inspired by [Severyn and Moschitti 2015], who proposed CNN-based re-rankers for Q&A. Our network consists of two subnetworks, each with 4-channels with kernel sizes $n = (\text{mask}(3), 3, 5, 8)$, and an overlap layer. We will now describe the main components:

5.2.1 Multi-Channel Convolution

The first subnetwork consists of only convolution kernels and aims to extract n-gram or keyword features from the caption sequence.

The convolution is applied over a sequence to extract n-gram features from different positions. Let $X \in \mathbb{R}^{L \times D}$ be the sentence matrix, where L is the sentence length, and D is the dimension of the i-th word in the sentence. Let also denote by $g \in \mathbb{R}^{nD}$ the kernel for the convolution operation, where n is the size of the kernel. For each i-th position in the sentence, w_i is the concatenation of n consecutive words as:

$$w_i = [x_i \oplus x_{i+1} \oplus \cdots \oplus x_{i+n-1}] \tag{5.1}$$

Our architecture uses multiple such kernels to generate feature maps m_i. The feature map for each window vector w_i can be written as:

$$m_i = f(w_i \cdot g + b) \tag{5.2}$$

where f is a nonlinear function, in our case we apply Relu function [Nair and Hinton 2010], and b is a bias. By performing this convolution for all windows w_i,

$i = \{1, \ldots, L-n+1\}$ of the sentence, we obtain the corresponding feature map for the n-gram:

$$\mathbf{m}_{\mathrm{n}} = [m_1 \oplus m_2 \oplus \ldots \oplus m_{L-n+1}] \tag{5.3}$$

We generate one such feature map for varying kernel sizes to capture the relationship between different n-grams of the sentence ($n = \{3, 5, 8\}$). For the single spotted word we apply a tri-gram kernel, masked on the word position (see details below). All these feature maps are finally concatenated (no pooling) and fed into the LSTM as shown in Figure 5.2.

5.2.2 Mask Convolution

Since we have only one candidate word at a time, we apply a convolution with *masking* in the candidate word side (first channel). Thus, the text hypothesis passes with the same padding (same length as the *max sequence*), though without the zeros, to the final joint layer. In this case, simply padding the sequence with zero has a negative impact on the learning stability of the network. The advantage of the mask is that it allows the network to learn short texts better and converge faster. Our Mask Convolution outperforms both standard CNN and Dynamic Convolutional (DCNN) [Kalchbrenner et al. 2014] at learning very short texts. We will discuss this comparison (*i.e.* standard CNN, DCNN, MaskCNN) in more detail in Section 5.4.3.

5.2.3 Multi-Channel Convolution-LSTM

Following the C-LSTM strategy proposed by [Zhou et al. 2015] we stack an LSTM after the convolution layer, whose hidden state features $\{h_1, \ldots, h_t\}$ capture the long dependencies in the sequence. Like the authors, we do not use a pooling operation after the convolution feature map. Pooling layer is usually applied after convolutions to extract the most relevant features in the sequence. However, by doing so we would be breaking the sequence ordering required in the input of the subsequent LSTM. Similarly, for the Multi-Channel Convolution model described above, we also learn the extracted word sequence n-gram without any feature selection or pooling operation.

5.2.4 Attention Mechanism

Attention-based models have shown promising results on various NLP tasks [Bahdanau et al. 2014]. Such a mechanism learns to focus on a specific part of the input (*e.g.* a relevant word in a sentence).

Attention mechanisms provide the model with direct access between states at a different point in time. Bahdanau et al. (2014) introduces an attention model that computes the context vector c_t as the weighted mean of h state sequence, given the model the hidden state h_t at each time step:

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_{tj} \tag{5.4}$$

where α_{ti} is the weight computed at each time t step for hidden state h_j, T is the number of time steps for the input sequence. The c context vector is used to compute the new state sequence s, where s_t depends on previous state s_{t-1}. The α_{tj} weight are then computed as:

$$e_{tj} = a\left(s_{t-1}, h_{tj}\right), \alpha_{tj} = \frac{\exp\left(e_{tj}\right)}{\sum_{k=1}^{T} \exp\left(e_{tk}\right)} \tag{5.5}$$

where the learned function $a(\cdot)$ can be considered as computing a scalar value of the important of h_j given the previous state s_{t-1}. This attention formulation has proven effective in different tasks, such as machine translation [Bahdanau et al. 2014], text spotting [Ghosh et al. 2017] and image captioning [Xu et al. 2015].

We apply an attention mechanism [Raffel and Ellis 2015] via an LSTM that captures the temporal dependencies in the sequence. This attention uses a Feed Forward Neural Network (FFN) attention function:

$$e_t = \tanh\left(h_t W_a\right) v_a^{\mathrm{T}} \tag{5.6}$$

where W_a is the attention of the hidden weight matrix and v_a is the output vector. As shown in Figure 5.2, the vector c is computed as a weighted average of h_t, given by α (defined below). The attention mechanism is used to produce a single vector c for the complete sequence as follows:

$$e_t = a\left(h_t\right), \alpha_t = \frac{\exp\left(e_t\right)}{\sum_{k=1}^{T} \exp\left(e_k\right)}, c = \sum_{t=1}^{T} \alpha_t h_t \tag{5.7}$$

where T is the total number of steps and α_t is the computed weight of each time step t for each state h_t, $a(\cdot)$ is a learnable function that depends only on h_t. Note

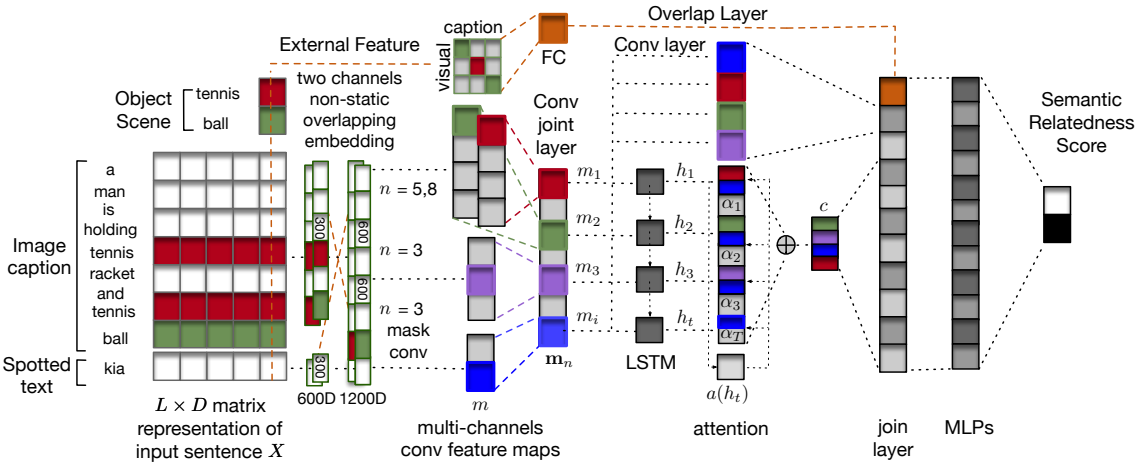

Figure 5.2: Model architecture, an end-to-end fashion post-process re-ranks potential words based on their semantic relatedness to the context in the image (objects, scenarios, natural language descriptions, ...).

that $a(\cdot)$ in [Bahdanau et al. 2014] as seen in Equation 5.5 depends on h and on the previous state s_{t-1} as $a(s_{t-1}, h_t)$. This setting permits the new state sequence s to have direct access to the entire sequence h. However, we are interested in integrating information over time rather than learning the sequence. Therefore, our $a(\cdot)$ depends only on h_t, since this formulation allows the attention layer to generate a context vector c by computing the adaptive weighted average of h the state sequence.

Since this attention computes the average over time, it discards the temporal order, which is ideal for learning semantic relations between words. By doing this, the attention gives higher weights to the most important words in the sentence without relying on sequence order.

5.2.5 Overlap Layer Dictionary

The main idea of the overlap layer is to emphasize the most important visual information occurring in the image by counting the overlap visual information (object, place, and caption). As shown in Figure 5.2 *tennis* occurs three times and *ball* twice, which indicates that the most import visual in that image is *tennis match*.

The overlap layer is just a frequency count dictionary to compute overlapping information in the inputs. The idea is to give more weight to the most frequent visual element, specially when it is observed by more than one visual classifier. The dictionary output is a fully connected layer. Tables 5.1 and 5.2 show a sample from the overlap information among the inputs.

Table 5.1: Exploiting overlapping information. Different examples from the dataset. The text hypotheses come from existing Text Spotting baselines and the visual context information is obtained from out-of-the-box computer vision classifiers (see Section 3.3). **Boldface** fonts show word overlapping among different classifiers. The overlapping (co-occurrence of words) is automatically computed and incorporated into the whole model via a fully connected layer that increases the weight of these words.

Text hypothesis	Object	Place	Caption
502	hotel	room	a woman laying on a bed with blanket
9h-18h	moped	**parking**	a close up of a **parking** meter with a sign on it
dell	**book**shop	**book**store	a woman sitting at a table with a laptop
snowbird	**ski**	**ski**	a man is **ski**ing down a hill on a **snow**y day
coke	**plate**	**pizz**eria	a table with a **pizza** and a fork on it **plate**
wii	**remote**	room	a close up of a **remote** control on a table

5.2.6 Multichannel Embedding Overlapping

In computer vision, images can have several channels (e.g. RGB channels). However, in the case of text, multiple channels could translate a different representation of the same input text, such as different word vectors for a word. Multichannel embedding can be either static or non-static (trainable). Kim (2014) show a mixed result, with slight improvement in accuracy, when comparing multichannel embedding and single or non-static embedding. In our case, we combined the two approaches (multichannel and non-static), and we introduced a dual-channel non-static embedding overlapping layer. Each input embedding channel overlaps with another input embedding. The idea is to overcome the limitations of train deep models without accessing a large amount of data and to prevent overfitting. Also, the model has access to information twice in a different order (overlapping). Thus, the convolution can have more overlapping pairs from both directions. This approach simulates the idea of Bi-directional LSTM but in one direction with a convolution.

5.2.7 Implementation Details

We merge all subnetworks described above (Attention, Overlap dictionary, etc.) into a joint layer that is fed into a loss function, which calculates the semantic relatedness of both inputs. We call the combined model Fusion Dual Convolution-LSTM-Attention ($\text{FDCLSTM}_{\text{AT}}$).

We concatenate the CNN outputs with the additional feature into MLP layers, and finally into a sigmoid layer performing binary classification. We trained

Table 5.2: Frequent text-object co-occurrences in COCO-text-Visual dataset.

Word	Visual	#	Word	Visual	#	Word	Visual	#
st	street	527	way	street	316	stop	street	1060
delta	airliner	169	kia	tennis	233	w	racket	149
ave	street	119	pizza	dining	44	starbucks	laptop	100
airways	airliner	83	bus	passenger	39	tennis	racket	72
wii	remote	35	apple	grocery	27	dell	desktop	22
heineken	refrigerator	11	rain	umbrella	16	cafe	bakery	5
25	baseball	14	puzzles	toyshop	5	p $	parking	4

the model with a binary cross-entropy loss (l) where the target value (in $[0, 1]$) is the semantic relatedness between the word and the visual. Instead of restricting ourselves to a simple similarity function, we let the network learn the margin between the two classes –*i.e.* the degree of similarity. For this, we increase the depth of the network after the MLPs merge layer with more fully connected layers. The network is trained using Nesterov-accelerated Adam (Nadam) [Dozat 2016], since it yields better results (especially in cases such as word vectors/neural language modeling) than other optimizers using only classical momentum (ADAM). We apply batch normalization (BN) [Ioffe and Szegedy 2015] after each convolution, and between each MLPs layer. We omitted the BN after the convolution for the model without attention (FDCLSTM), since BN deteriorated the performance. Additionally, we consider a 70% dropout [Srivastava et al. 2014] between each MLPs for regularization purposes.

5.3 Experiments

To validate the effectiveness of our model, in this section we compare our pipeline with several of the most recent state-of-the-art word and sentence level models: 1) pre-trained, 2) tuned, or 3) trained on the same dataset. Furthermore, we combined a mixture model of experts including a word and sentence joined model. Data and preliminaries are discussed next, baselines information in Section 5.3.4, and results in Section 5.4.

5.3.1 Dataset

We evaluate our pipeline on a COCO based dataset [Lin et al. 2014], for training COCO-text [Veit et al. 2016], and for evaluating ICDAR17 [Gomez et al. 2017], which are described in Section 3. We train our model on COCO-text-Visual dataset

that consists of 16k images with associated bounding boxes, and a 60k textual visual information (*i.e.* caption, object and scene). For evaluation, we test our model on ICDAR17-Task3-V, a dataset that is based on the ICDAR17 Task 3 end-to-end text recognition. Similar to the COCO-text-Visual dataset, we only introduce the visual context information (textual dataset) for each bounding box. It consists of 10k images with 25k textual data for test and validation.

5.3.2 Preliminaries

As in Chapter 4 we follow the same restrictive protocol than the standard one proposed by [Wang et al. 2011] and adopted in most state-of-the-art benchmarks, which do not consider words with less than three characters or with non-alphanumeric characters. We consider all cases in the dataset, and words with less than three characters are also evaluated.

Note that this standard protocol proposed by [Wang et al. 2011] was introduced to overcome the problem of false positives with short words, which most current state-of-the-art approaches struggle with, including our baseline.

As in Chapter 4 we used two pre-trained deep models, CNN [Jaderberg et al. 2016] and LSTM [Ghosh et al. 2017] as a baseline (BL) to extract the initial list of word hypotheses. Since these BLs need to be fed with the cropped words, when evaluating on the ICDAR-2017-Task3 dataset [Gomez et al. 2017], we use the ground truth bounding boxes of the words.

5.3.3 Methods Using Word Level Information

First, we compare our result with current state-of-the-art word embeddings trained on a large general text -*i.e. w2v* and *fasttext*, etc. which are described in Chapter 4. The word model uses only objects and places information and ignored the caption. The comparison results are shown in Table 5.4 word-level section. Table 5.3 shows Chapter 4 results plus word-level experiment results with recent sentence encoder or context-aware dynamic embedding pre-trained models USE-T and BERT. However, the word-level similarity is worse than the static embeddings, such as Glove.

5.3.4 Methods Using Sentence Level Information

In this section, we employ the image description (*i.e.* caption) as sentence level information to assess the semantic relation with the text hypothesis. In particular, we

Table 5.3: Comparison of different models in a word level re-ranking scenario. The ♠ indicates Chapter 4 result.

Model	CNN	k	list	LSTM	k	list
Baseline	19.7	-	-	17.9	-	-
BL+Word2vec [Mikolov et al. 2013b]	21.8	5	80.0	19.5	4	76.9
BL+Glove [Pennington et al. 2014]	22.0	7	75.8	19.1	4	75.3
BL+Fasttext [Joulin et al. 2017]	21.9	7	75.4	19.4	4	76.1
BL+Sw2v [Mancini et al. 2017]	21.8	7	75.2	19.4	4	76.4
BL+USE-T$_w$ [Cer et al. 2018]	21.9	6	77.9	19.1	4	76.3
BL+LSTMEmbed [Iacobacci 2019]	21.6	7	73.7	19.2	4	75.8
BL+BERT$_w$ [Devlin et al. 2019]	21.8	6	77.5	18.9	4	74.7
BL+RWE-Ft [Camacho et al. 2019]	22.0	7	75.8	**19.5**	4	77.1
BL+TWE ♠	**22.2**	7	76.3	19.5	4	76.7

compare the results of our encoder with several state-of-the-art sentence encoders, tuned or trained on the same dataset. We use the cosine distance to compute the similarity between an automatically generated image caption and the candidate word. Word-to-sentence representations are computed with: USE-T [Cer et al. 2018], BERT [Devlin et al. 2019], and Infersent [Conneau et al. 2017a] with Glove [Pennington et al. 2014]. The remaining of the systems in Table 5.4 are trained in the same conditions and on the same dataset as our model, with Glove initialization and dual-channel overlapping non-static pre-trained embedding. In the next section, we describe each sentence encoder baseline in more detail.

5.3.4.1 Sentence Level Baselines

Here, we will compare the results of our FDCLSTM models against other sentence level encoders that are, tuned/trained on the same dataset, as shown in Table 5.4.

Attentive LSTM: [Tan et al. 2016] QA-LSTM is a siamese based network that uses word-level attention to deliver more weight to certain words in the input sentences. Therefore, the computed weight takes into consideration the information from the input sentence in a word-level manner. Since our goal is the semantic relatedness score, we discard the similarity function proposed by the author and let the network learn the degree of similarity in an end-to-end fashion. This architecture is very similar to ours but with n-gram level features, since the CNN is used as keyword detector.

MVCNN: [Yin and Schütze 2016] The multichannel variable-size convolution network is a model that extracts multiple phrases with a variable size kernel. MVCNN

combines multiple versions of pre-trained embedding to extract extra information from the same set of word vectors. Although its approach uses multiple embedding, we only employ the two richest word embeddings in our model. (*e.g.* word2vec [Mikolov et al. 2013b] and Glove [Pennington et al. 2014])[1]. We also use the Mutual-Learning trick to maintain the same words across each embedding during training.

CNN+RNN: [Wang et al. 2016a] Multichannel convolution layers stacked into an RNN. The RNN is combined to extract the local features generated by the CNN, and to learn the long-distance dependencies in the sequence. In our case, the RNN is trained to process the caption while the CNN with a pooling layer is used to extract the most important n-gram feature from the sequence.

C-LSTM: [Zhou et al. 2015] This architecture is similar to CNN+RNN except that C-LSTM discards the pooling layer. The C-LSTM without pooling learns the high-level extracted key word or phrases representations directly, without any feature downsampling. Our work is inspired by this architecture.

InferSent: [Conneau et al. 2017a] is a bi-directional LSTM with max-pooling. The network computes the n-vectors for the n-word and each output vector is a concatenation of LSTM forward pass and backward pass. The max-pool is applied after each vector concatenation to form the final vector. We obtain the word vector representation for each encoder to compute the cosine distance. Our task relies more on learning the n-gram feature and thus, a tuning or training bi-directional LSTM model that learns the sequence in bi-directional order performs poorly.

USE-Transformer: [Cer et al. 2018] Universal Sentence Encoder (USE) is the current state-of-the-art in Semantic Textual Similarity (STS). This work proposed two USE models with different encoder architectures to achieve distinct design goals. The first is based on the USE-T transformer architecture [Vaswani et al. 2017], which targets with high accuracy at the cost of complexity and resource consumption. The second is based on a Deep Averaging Network that targets efficient inference with slightly reduced accuracy. We experimented with USE-T fine tuning and feature extraction to compute the semantic relation with the cosine distance. We extended our experiment to word-to-word to be able to compare with different word-to-word methods, as shown in Table 5.3.

[1] Adding more embeddings has no positive impact on accuracy.

Transformer: [Vaswani et al. 2017]: We also considered training the transformer from scratch on the same dataset. Transformer utilizes a self-attention mechanism to compute representations of its input and output without using sequence-alignment networks such as RNNs. We train a dual transformer based encoder that initialized with Glove [Pennington et al. 2014]. Each encoder has two multi-head attention heads followed by a pooling layer. We concatenate both pooling layers into a MLP joint layer with 10% dropout and finally a sigmoid layer. The advantage of the transformer is that it uses a multi-head attention mechanism that allows it to model long dependencies regardless of their distance from the target output or input sentence. However, since there is not enough data, and only relies on a short sentence (caption), the self-attention mechanism is unable to learn the correlation between words.

BERT [2]**:** [Devlin et al. 2019] (Bidirectional[3] Encoder Representations from Transformers) has produced groundbreaking results in many tasks, including question answer and natural language inference. However, according to its main authors, it is not suited to Semantic Textual Similarity (STS) tasks, since it does not generate a meaningful vector to compute the cosine distance as seen in Table 5.4, row *BERT-feature*. We therefore fine-tuned the model with one layer to compute the semantic score between a caption and a candidate word. In particular, we fed the sentence representation into a linear layer and a softmax for sentence pair tasks (QA re-ranking task). A fine-tuned BERT outperforms our model when applied on the CNN baseline. However, with the second baseline (lexicon-free LSTM), our model without attention outperforms both attention models ($\text{FDLSTM}_{\text{AT}}$ and BERT). Applying an attention mechanism results in an incorrect correlation between words (candidate text and visual context object) since there are many images in which the spotted text has a relationship with the context. Although BERT achieves slightly better results than our model on the lexicon-based CNN, our model has a much lower training cost, and can be trained from scratch on any specific domain.

5.3.5 Effect of Unigram Probabilities

The use of a language model leads to significant improvements in text spotting when the data is too small for a DNN. Ghosh et al. (2017) show that the language

[2]We use the basic BERT-base-uncased model

[3]The transformer encoder handles the entire sequence at once (left-to-right/right-to-left). Therefore, although it is refereed to as bidirectional, it could be regarded as a non-directional encoder.

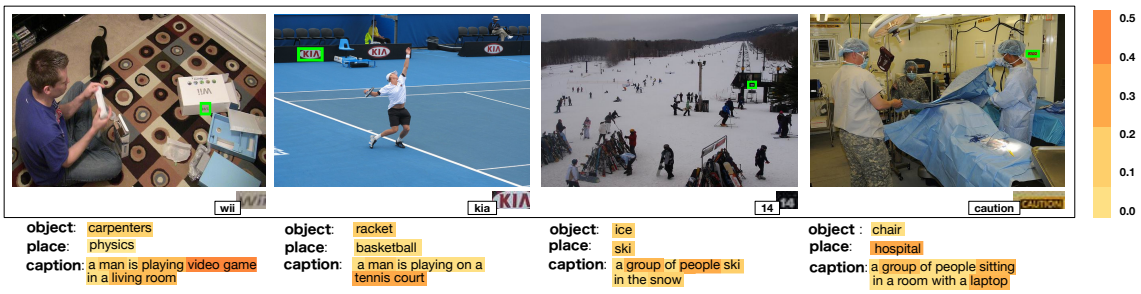

Figure 5.3: Examples illustrate that natural language description overcomes the limitation of visual contexts, such as object and scene, and provide more visual context information such as action, location, and specific details.

model improves accuracy by around 8%. However, the problem of a fixed lexicon is the persistence of out-of-vocabulary words (OOV). Therefore, we use the unigram frequencies ULM (7M fixed lexicon) with OOV smoothing after FDCLSTM to re-rank out false-positive short words. As seen in Table 5.4 $\text{FDCLSTM}_{\text{AT+lexicon}}$ achieves the best results when adding lexicon with FDCLSTM.

5.4 Results

This section compares the performance of our system with the standard benchmarks and state-of-the-art systems. In particular, we use different *similarity* or *relatedness* scorers to reorder the k-best hypothesis produced by an off-the-shelf state-of-the-art text spotting system. We experimented extracting and re-ranking k-best hypotheses for $k = 1 \ldots 10$.

Similar to Chapter 4, we use two pre-trained deep models: a CNN [Jaderberg et al. 2016] and an LSTM [Ghosh et al. 2017] as baselines (BLs) to extract the initial list of word hypotheses. Since these BLs need to be fed with cropped words (*i.e.* image regions containing only text), when evaluating on the ICDAR-2017-Task3 dataset we will use the ground truth bounding boxes of the words. Thus, we are not evaluating a whole end-to-end system (bounding box detection and recognition), but only the influence of adding external natural language understanding knowledge to the second stage (recognition). In other words, each baseline takes a text image bounding box (bb) as input and produces k candidate words $w_1 \ldots w_k$ plus a probability for each prediction $P(w_i|bb) \quad i = 1 \ldots k$.

The CNN baseline uses a closed lexicon and therefore it cannot recognize any word outside its 90k-word dictionary. Table 5.4 presents four different accuracy metrics for this case: 1) **full** columns correspond to the accuracy on the whole

Table 5.4: Best results after re-ranking using different re-rankers, and different values for the k-best hypotheses extracted from the baseline output (%). In addition, to evaluating our re-ranker with MRR, we fixed k $\text{CNN}_{k=8}$ $\text{LSTM}_{k=4}$. ♠ marks results presented in Chapter 4, and ♢ indicates the results in this chapter.

Model	**CNN**					**LSTM**			
	full	*dict*	*list*	*k*	MRR	*full*	*list*	k	MRR
Baseline (BL)	**full: 19.7 dict: 56.0**					**full: 17.9**			
Word Level									
BL+Word2vec [Mikolov et al. 2013b]	21.8	62.1	80.0	5	44.3	19.5	76.9	4	80.4
BL+Glove [Pennington et al. 2014]	22.0	62.5	75.8	7	44.5	19.1	75.3	4	78.8
BL+Sw2v [Mancini et al. 2017]	21.8	62.1	75.2	7	44.3	19.4	76.4	4	80.1
BL+Fasttext [Joulin et al. 2017]	21.9	62.2	75.4	7	44.6	19.4	76.1	4	80.3
BL+RWE-w2v [Camacho et al. 2019]	21.9	62.4	75.6	7	44.5	19.6	77.3	4	80.7
BL+RWE-fasttext [Camacho et al. 2019]	22.0	62.5	75.8	7	44.6	19.5	77.1	4	80.4
BL+LSTMmebed [Iacobacci 2019]	21.6	60.8	73.7	7	44.0	19.2	75.8	4	79.6
BL+TWE ♠	22.2	63.0	76.3	7	44.7	19.5	76.7	4	80.2
Sentence Level									
BL+C-LSTM [Zhou et al. 2015]	21.4	61.0	71.3	8	45.6	18.9	74.7	4	80.7
BL+CNN-RNN [Wang et al. 2016a]	21.7	61.8	73.3	8	44.5	19.5	77.1	4	80.9
BL+MVCNN [Yin and Schütze 2016]	21.3	60.6	71.9	8	44.2	19.2	75.8	4	78.8
BL+Attentive LSTM [Tan et al. 2016]	21.9	62.4	74.0	8	45.7	19.1	71.4	5	80.2
BL+InferSent [Conneau et al. 2017a]	22.0	62.5	75.8	7	44.5	19.4	76.7	4	79.7
BL+Transformer [Vaswani et al. 2017]	21.7	61.6	73.1	8	44.3	19.2	64.6	9	77.9
BL+USE-T [Cer et al. 2018]	22.0	62.5	78.3	6	44.7	19.2	75.8	4	79.5
BL+BERT-feature [Devlin et al. 2019]	21.7	61.6	74.6	7	45.0	19.3	76.2	4	81.2
BL+BERT (fine-tune) [Devlin et al. 2019]	**22.7**	64.6	76.6	8	**45.9**	20.1	67.0	9	79.1
BL+FDCLSTM ♢	22.3	63.3	75.1	8	45.0	**20.2**	67.9	9	79.8
$\text{BL+FDCLSTM}_{\text{AT}}$ ♢	22.4	63.7	75.5	8	**45.9**	20.1	67.6	9	**81.8**
$\text{BL+FDCLSTM}_{\text{Lexicon}}$ ♢	22.6	64.3	76.3	8	45.1	19.4	76.4	4	78.8
$\text{BL+FDCLSTM}_{\text{AT+Lexicon}}$ ♢	22.6	64.3	76.3	8	45.1	19.7	77.8	4	80.4
Combined Model with FDCLSTM (ours)									
$\text{BL+FDCLSTM}_{\text{AT}}$+InferSent [Conneau et al. 2017a]	22.7	64.5	78.1	7	45.2	19.7	77.5	4	80.0
$\text{BL+FDCLSTM}_{\text{AT}}$+USE-T [Cer et al. 2018]	22.8	64.8	78.5	7	45.4	19.3	76.2	4	79.2
$\text{BL+FDCLSTM}_{\text{AT}}$+BERT [Devlin et al. 2019] ♢	22.0	62.5	74.2	8	43.5	**20.3**	69.7	8	77.9
$\text{BL+FDCLSTM}_{\text{AT}}$+TWE ♢ ♠	**23.0**	**65.2**	79.0	7	45.4	19.8	78.2	4	80.7

dataset. 2) **dict** columns correspond to the accuracy on the cases where the target word is among the 90k-words in the CNN dictionary (which account for 43.3% of the whole dataset. 3) **list** columns report the accuracy on the cases where the right word was among the k-best produced by the baseline. 4) **MRR** stands for Mean Reciprocal Rank, which is computed as $\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$, where Q is the number of images in the dataset, and rank_i is the position of the correct answer in the re-ranked hypothesis list for image i.

Table 5.4 compares results of our model and its variants (*i.e.* attention, lexicon or combined with other model) with other baselines in **sentence level** and **word-level**. $\text{FDLSTM}_{\text{AT}}$+TWE shows the performance of the best combination of both. Overall, our combined model gets the best result, outperforming all baselines. Fig-

ure 5.3 illustrate that adding natural language description overcome the limitation of visual context, such as object and scene, and provide more visual context information such as action, location and specific details.

On the other hand, our model FDCLSTM without attention achieves a better result in the case of the second baseline LSTM, which generates many false-positives and short words. The advantage of the attention mechanism is the ability to integrate information over time, and it allows the model to refer to specific points in the sequence when computing its output. However, in this case, the attention attends the wrong context, since there are many words with no correlation with the context, or that do not correspond to actual words in the language.

The USE-T model seems to require a shorter hypothesis list to get top performance when the right word is in the hypothesis list. A fine-tuned BERT, on the same dataset outperforms our model BL+FDCLSTM$_{\text{AT+lexicon}}$ by a small non-significant margin in the first baseline.

5.4.1 Combining Sentence and Word Level

As in Chapter 4 we combined different re-rankers (or *experts*) [Hinton 1999]. Product of Expert (PoE) takes advantage of each expert and can produce a much stronger model for finding the candidate word with highest probability after re-ranking. The combined probability of a given word $\mathbf{w}$ can be written as:

$$\arg\max_{\mathbf{w}} P\left(\mathbf{w}|\theta_1 \dots \theta_n\right) = \arg\max_{\mathbf{w}} \frac{\Pi_m p_m\left(\mathbf{w}|\theta_m\right)}{\sum_{\mathbf{c}} \Pi_m p_m\left(\mathbf{c}|\theta_m\right)} \tag{5.8}$$

$$= \arg\max_{\mathbf{w}} \Pi_m p_m\left(\mathbf{w}|\theta_m\right) \tag{5.9}$$

where, in our case, $p_m\left(\mathbf{w}|\theta_m\right)$ are the probabilities assigned by each expert to the candidate word $\mathbf{w}$. Note that since we are just interested in retrieving the candidate word with higher probability, we do not need to normalize as in Equation 5.9. Table 5.5 shows the results using different combinations of experts, such as one of the baselines, our proposed model FDCLSTM, TWE embedding, BERT, etc.

As shown in Table 5.4 the combination of our model with other experts produces higher accuracy and a better retrieval score, since our model combined with the TWE model (Section 4.3.3) boosts the accuracy up to 3.3% in the case of the CNN. In contrast, the PoE on the second baseline struggles with false positives, such as joint characters not corresponding to an actual word (*e.g. lemid, crucifor,*

Table 5.5: Best results after combining different **experts** with our model for re-ranking k-best hypotheses extracted from the baseline output (%). In addition, to test our re-ranker with MRR, we fixed k $\text{CNN}_{k=8}$ $\text{LSTM}_{k=4}$. The ♢, ♠ indicates the result of this Chapter and of the work presented in Chapter 4, respectively. Table 5.4 shows the full comparison with other models.

Model	**CNN**					**LSTM**			
	full	*dict*	*list*	*k*	MRR	*full*	*list*	k	MRR
Baseline (BL)	**full: 19.7 dict: 56.0**					**full: 17.9**			
Combined Model with FDCLSTM (ours)									
BL+$\text{FDCLSTM}_{\text{AT}}$+InferSent [Conneau et al. 2017a]	22.7	64.5	78.1	7	45.2	19.7	77.5	4	80.0
BL+$\text{FDCLSTM}_{\text{AT}}$+USE-T [Cer et al. 2018]	22.8	64.8	78.5	7	45.4	19.3	76.2	4	79.2
BL+$\text{FDCLSTM}_{\text{AT}}$+BERT [Devlin et al. 2019] ♢	22.0	62.5	74.2	8	43.5	**20.3**	69.7	8	77.9
BL+$\text{FDCLSTM}_{\text{AT}}$+TWE ♢ ♠	**23.0**	65.2	79.0	7	45.4	19.8	78.2	4	80.7
Combined Model with Bert									
BL+BERT+InferSent [Conneau et al. 2017a]	22.6	64.1	**86.6**	5	45.1	19.5	**89.0**	4	79.3
BL+BERT+USE-T [Cer et al. 2018]	22.5	63.8	77.3	7	45.3	19.4	88.5	4	79.0
BL+BERT+TWE ♠	22.7	64.6	76.6	8	45.1	19.9	76.4	9	78.4

etc.), and the combined experts did not add any additional improvement. However, by combining our model with BERT [Devlin et al. 2019] we achieved a slight improvement of 2.4%.

BERT experiment. We also apply the same experiment with BERT, although no accuracy improvement was obtained, as shown in Table 5.5. Since BERT is trained on huge amounts of data, its output semantic relatedness score is either very high or very low, which is not convenient for re-ranking. On the other hand, our model gives each word a moderate confidence score, which can be further re-ranked by adding more visual context information. In other words, our approach only modifies (re-rankes) the individual small word scores based on the presence on the visual context and is therefore unable to re-rank a confident classifier like BERT.

5.4.2 Evaluation with other Text Spotting Experts

In this section, we evaluate our $\text{FDCLSTM}_{\text{AT}}$ re-ranker model with regard to the output of recent state-of-the-art text spotting systems. To do so, we add two more baselines, CNN-LM [Fang et al. 2018] and CRNN [Shi et al. 2016] apart from the CNN and LSTM used in the previous experiments.

Convolutional Recurrent Neural Network (CRNN) [Shi et al. 2016]. The first baseline is a CRNN that can directly learn the words from sequence labels, without relying on characters annotations. The encoder uses a CNN to extract a set of features from the image. Specifically, the network uses CNN without fully connected layers to extract sequential feature representation from an input image,

Table 5.6: Best results after combining different Text Spotting systems with our model FDCLSTM$_{AT}$ for re-ranking k-best hypotheses extracted from different state-of-the-art baselines (%). The ♢, ♠ indicates this chapter results and Chapter 4 work, respectively.

Model	full	list	k
Baseline	CNN-LM: **26.2** CRNN: 21.1 CNN: 19.7, LSTM : 17.9		
TWE ♠	25.34	29.7	4
BERT (fine-tune)	26.52	**55.9**	2
FDCLSTM$_{AT}$ ♢	25.57	30.0	4
BERT+TWE ♠	25.79	30.3	4
BERT+FDCLSTM$_{AT}$ ♢	26.40	55.7	2
FDCLSTM$_{AT}$+TWE ♢ ♠	**26.57**	31.2	4

which are then fed into a bidirectional RNN. A Connectionist Temporal Classification [Graves et al. 2006] based method is used to convert the per-frame predictions made by RNN into a label sequence.

CNN-Attention with (CNN-LM) [Fang et al. 2018]. The second baseline is the most recent state-of-the-art model, and like our LSTM baseline, also produces the final output words in the form of probable character sequences without any fixed lexicon. The model is based on a CNN encoder-decoder with attention and a CNN character language model. The final character prediction is a element-wise addition of the attention plus the language vector. The network is trained in an end-to-end fashion with multiple losses (attention and language model).

Implementation Details. Since CRNN and CNN-LM do not produce a list of hypotheses, but rather a single word as output, we use a list of $k = 4$ hypothesis. These consist of the top prediction from of each of the old baselines (CNN and LSTM) and the single word output by each of the new baselines (CRNN and CNN-LM). That is, we are ranking the hypotheses generated by four different systems instead of re-ranking the output of a single system.

We apply our best proposed model and the combined re-ranker in Table 5.4 (*i.e.* FDCLSTM$_{AT}$, TWE, BERT and combined model with FDCLSTM) to the $k = 4$ hypothesis list. The output of the weakest LSTM baseline is filtered with a unigram language model to remove uncommon and non-words.

In previous experiments, our model was used to re-rank a softmax output, and thus the re-ranker had some prior probability distribution to work on. In this case, since each hypothesis comes from a different baseline, we do not have a softmax

Table 5.7: Comparison of different word level attention models with MaskCNN.

model	CNN	LSTM
Baseline	19.7	17.9
BL+Attentive LSTM [Tan et al. 2016]	21.9	19.1
BL+Attentive DCNN [Kalchbrenner et al. 2014]	21.1	19.3
BL+Attentive MaskCNN (ours)	**22.3**	**19.8**

distribution, and we back off to a uniform prior; thus, we lose some valuable information. Despite this, we achieved a slight improvement over CNN-LM [Fang et al. 2018], as shown in Table 5.6. Even though the improvement is not statistically significant, we believe that this result indicates that our approach can contribute to improving results even in top state-of-the-art systems. Note that our combined model performs better than a single model, while BERT is the best reranker when used alone. We discuss this limitation in more detail (*i.e.* re-ranking with a uniform prior) in the discussion Section 5.6.

5.4.3 Validation of Mask Convolution Layer : A Case Study

In order to assess the performance of the introduced Mask Convolution, we carried out a number of validation experiments, in which this layer was introduced in existing state-of-the-art architectures. We propose an **Attentive MaskCNN** based on Attentive-LSTM [Tan et al. 2016] that uses LSTM with word-level attention. In particular, we use the same architecture as in Figure 5.2 but we replace the Convolution-LSTM$_{AT}$ with Attentive-LSTM architecture. We keep the **overlap layer** and **embedding overlapping** in both experiments. In our model, we replace the LSTM with a Mask Convolution with kernel size $k = 12$ to be able to process the whole caption. Both networks are trained on the visual context dataset with the same procedure as the FDCLSTM$_{AT}$ network. We also compared our model with Dynamic Convolutional [Kalchbrenner et al. 2014] (DCNN), that uses a feature graph to capture textual information among texts of variable lengths. DCNN uses dynamic k-pooling which, unlike local max-pooling, outputs a k-max value of different convolutional dimensions (sentences with different length). As shown in Table 5.7 our model achieves a better result than DCNN and the original work in this task and can be parallelized for faster training. Our proposed Attentive-MaskCNN overcomes some of the limitations of LSTM with attention mechanisms, such as slow training and complexity.

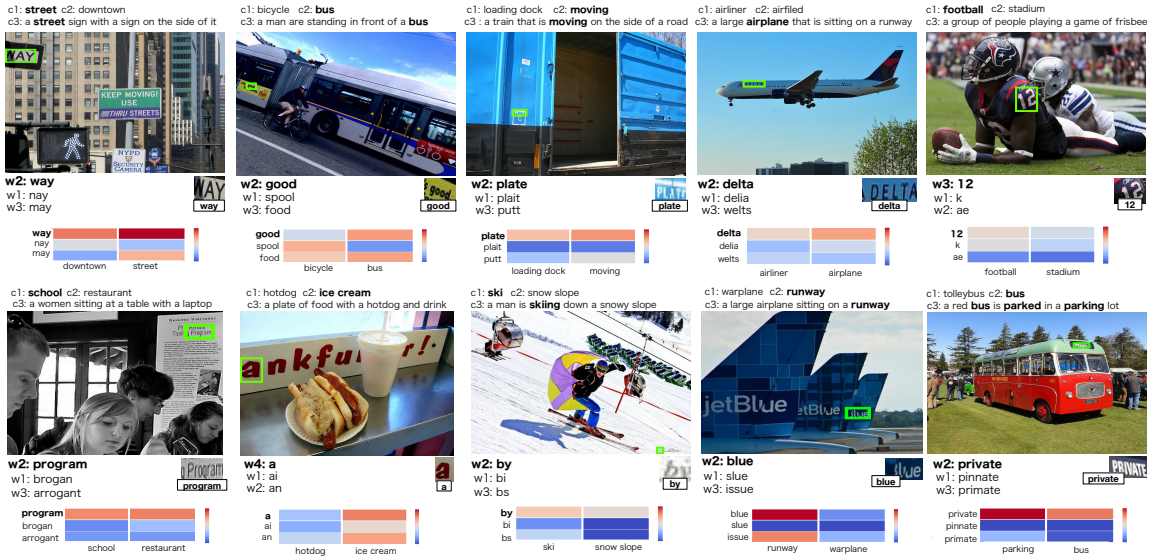

Figure 5.4: Examples of candidate re-ranking using (c1) object, (c2) place, and (c3) caption information. The top three examples are re-ranked based on a semantic relatedness score. The *delta-airliner*, which frequently co-occurs in training data, is captured by overlap layers. The term *12-football* shows a relation between sport and numbers. Also, *program-school* has a much more distant relation but our model is able to re-rank the most closely related words. The terms *blue-runway* and *bus-private* show that the overlap layer can be effective when the visual context appears in more than one visual context classifier. Finally *hotdog-a* and *by-ski* have no semantic correlation but are solved by the network thanks to the frequency count dictionary.

5.5 General Text Experiment : A Case Study

This section evaluates our method on general text as proof of concept that our model can be generalized. We evaluated our model with a dummy task on general text collected from Wikipedia. The task is to learn the semantic relatedness between the **title** and the **description** of a Wikipedia entry. We compared our model with most recent state-of-the-art semantic textual similarity method USE-T [Cer et al. 2018]. Although this architecture is designed to tackle a specific problem, as shown in Table 5.8, our model overperforms USE-T [Cer et al. 2018] in this task. Our model is trained on a 50k pair word-description from Wikipedia (*e.g.*, *Alaska – is a U.S. state located in the northwest extremity of North America*). However, we add a cosine similarity layer on the top of USE-T to compute the similarity score. This is just a confirmation of our hypothesis that the tasks require a broader perspective to determine whether two text fragments are related more closely than to determine whether they are similar as in traditional semantic similarity approach.

Table 5.8: General text experiment results.

Model	Recall	Precision	F-score
USE-T [Cer et al. 2018]	0.83	0.43	0.56
$FDCLSTM_{AT}$ (This chapter)	0.66	0.65	**0.66**

5.6 Discussion

Our proposed approach re-ranks candidate words based on their semantic correlation score with the visual context from the image. However, there are some cases when there is no direct semantic correlation or no relation at all between the visual context and the candidate word. We therefore proposed the inclusion of an overlap layer to address this limitation by learning correlations from the training dataset. For instance, as in Figure 5.4 and Table 5.2 the company name *Delta* and the visual context *airline*. Also, adding the unigram frequency helps to filter out short words and false positives as shown in the examples *a-hotdog* and *by-skiing*.

Although our model combined with unigram frequency yields the best results, there are some cases where high or low word counts impact the accuracy in a negative way. That is, high-frequency words can be over-ranked and selected before the right answer, and low-frequency words may be under-ranked, preventing the right choice from reaching the top position. This problem can be tackled by adjusting the weight of the common words in the unigram model. Also, it can be corrected for by adding the most common pairs such as a commercial Ad brand (*i.e.Nike-Sport*, *KFC-restaurant*) to the training dataset that is not semantically related in general text, but commonly co-occur in the real world. Note that in Chapter 4, to overcome word embedding limitations when the candidate word and the image objects are not semantically related in general text, but frequently co-occur in the dataset, we use training data to estimate the conditional probability $P(w|c)$ of a word w given that object c appears in the image.

We believe that BERT [Devlin et al. 2019] offers an advantage over our model in terms of the amount and diversity of the training data. For instance, it is able to solve cases without direct context such as *sex-street*, *7-food* and *anderson-plaza* as shown in Figure 5.5, the model is confident about this text-context correlation.

Also, since the text in images is not always related to its environment (*e.g.* a commercial ad for a popular soft drink may appear almost anywhere), this approach may help to resolve only a fraction of cases, but given its low cost, it may prove useful for domain adaptation of general text spotting systems. In addition,

Figure 5.5: Examples of false-positives by our model that a fine-tune BERT was able to solve. Our proposed language model is also able to solve some non-semantic context and frequency count dependent cases with the help of the overlap layer as shown in the last example *adam-child*. According to our best model FDCLSTM$_{\text{AT}+\text{Lexicon}}$ there is no semantic relation between the spotted word and its environmental context. However, as seen in the top examples the BERT attention indicates that there is a correlation between these words in the training data.

this approach is easy to train and can be used as a drop-in complement for any text-spotting algorithm that outputs a ranking of word hypotheses.

Limitation. The main limitation of this approach is that it depends on the baseline softmax output to re-rank the most closely related word. In particular, the semantic relatedness score suppresses unrelated words and boosts the most probably related word by simple dot product multiplication, as shown in Table 4.3 in Chapter 4 which includes a sample of the modified probabilities after adding the visual context. For instance, as seen in Figure 5.5, *spy-street* and *anderson-plaza*, the softmax score is very confident about its output and cannot be re-ranked.

5.7 Conclusion

In this chapter, we have proposed a simple deep learning architecture to learn the semantic relatedness of word-to-word and word-to-sentence pairs, and we have shown how it outperforms other semantic similarity scorers when used to re-rank candidate answers in the Text Spotting problem. In particular, these results prove that this introduction of contextual semantic information significatively increases the performance of the text spotting system up to 3.3 points in benchmark dataset.

Chapter 6

Summary and Future Work

In this chapter we review the achievements of the work presented in this thesis and also discuss potential directions for future research.

6.1 Achievements

In this thesis we have proposed several methods for improving text spotting system without any additional cost (*i.e.* re-training or tuning) based on the advanced capabilities of recent developments in natural language processing, such as word embeddings and deep learning frameworks. We have also contributed to the public dataset available for this problem, enriching it with image context information. Figure 6.1 presents a graphical abstract for the proposed methods in this thesis.

In Chapter 3, we proposed a textual visual context dataset for an existing image based dataset COCO-text [Veit et al. 2016]. We approached this in two stages, first by adopting out-of-the-box visual classifiers (object, scene, caption generator) to extract the visual context from each image. Then we used existing baseline text spotting systems for text hypothesis generation, which means that this dataset annotation can be improved in the future when better systems become available. Our main contribution to this dataset is the combined visual context dataset, which provides the unrestricted-OCR research community with the chance to use semantic relatedness between text and image to improve their results.

In Chapter 4, we proposed a word-level re-ranker framework for text spotting system. This builds upon the word embedding as a semantic relatedness re-ranker. The word-level re-ranker framework outperforms all existing general word embedding methods in this task. Also, another contribution in Section 4.3 is the presentation of similarity to probabilities transformation methods that are based on hypotheses revision [Blok et al. 2003].

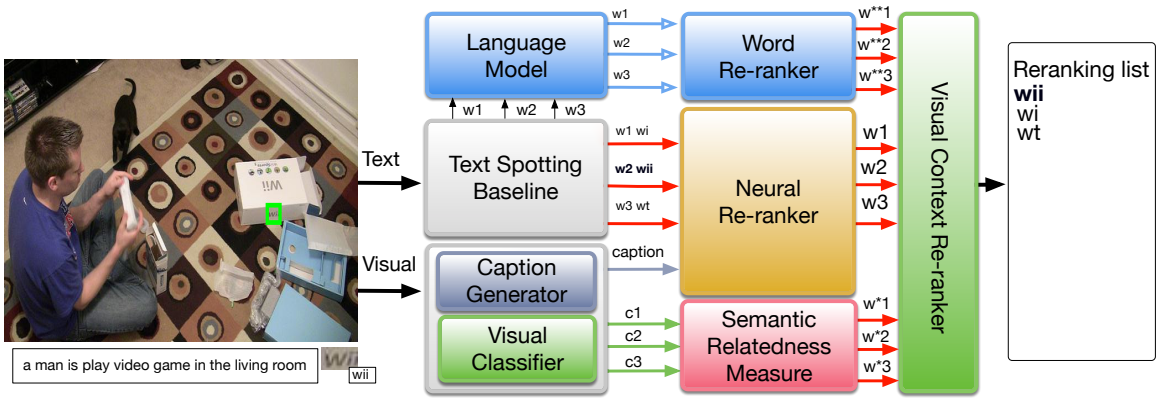

Figure 6.1: The overall architecture of the proposed visual context information pipeline integration into the text spotting system. This approach uses all the semantic relatedness measures from the image (objects (c_i), places (c_j), caption) to re-rank the word hypothesis.

Finally, in Chapter 5, we presented an end-to-end sentence-level visual based re-ranker. This builds upon the text modeling based pipeline, which was developed to explore the advantages of recent advances in text modeling and machine translation. The neural based re-ranker is based on a shallow-deep learning C-LSTM that takes the word and its all visual context, as in Chapter 4 (scene, object) and adds an image description, and poses the semantic relatedness score as a binary classification problem. Also, we presented our combined model based on Product of Expert [Hinton 1999], developed to take advantage of sharper distribution from each expert. This combined based re-ranker outperforms all previous methods (including Chapter 4 and BERT [Devlin et al. 2019]).

6.2 Conclusion

In this thesis, we have demonstrated the benefit of leveraging these NLP techniques into computer vision problems that boost the performance of the text spotting system up to 3.3 points on a benchmark dataset without tuning or training. Therefore, to answer our main question: **Could Word Semantic Relations be Beneficial for Scene Text Recognition?** We believe word semantics could be beneficial for text spotting but to a limited extent as there are only a fraction of cases where the word is related with its visual context. However, it may be useful for domain adaptation or general text spotting systems since (1) it has a low computational cost, (2) it uses off-the-shelf state-of-the-art tools, and (3) it can be used as

a drop-in complement for any text-spotting algorithm that outputs a ranking of word hypotheses.

Also, we show in this thesis that pre-trained contextual embedding systems like BERT constitute the better option for getting state-of-the-art results (though with higher computational cost), especially as a post-processing task. Furthermore, we point out another approach that can be used: Knowledge-based embeddings, or sense embeddings (though our preliminary results fail to show any improvement when using senses instead of words).

6.3 Future Work

This section discusses some potentially promising directions for future work.

Sense Embedding. It would be useful to explore further other types of word embedding for this task. Recently, a significant gain in semantic similarity was achieved by adding senses to traditional word embedding [Mancini et al. 2017, Iacobacci 2019]. In particular, adding senses or meanings of words, makes it possible to overcome a major limitation of word-embedding known as *Conflation Deficiency*. It ignores the fact that each word has multiple meanings and conflates all these multiple meanings into a single representation. However, as we show in this thesis in Chapter 4, having the sense of the word does not necessarily improve the results. Therefore, our primary approach is to combine the senses vector with the original vector of the word in the same semantic space. This preliminary experiment shows promising results in semantic similarity tasks. In particular, adding the senses to general word embedding results outperformed by both Glove [Pennington et al. 2014] and word2vec [Mikolov et al. 2013b] in this task. Also, we would like to employ contextualized embeddings like BERT to predict the sense of the word based on the visual context [Levine et al. 2019].

Expanded visual context data. Our dataset as described in Chapter 3 would benefit from better bounding box annotation in COCO-text (ideally manual annotation). It also would be useful to extend the visual context provided (currently two objects, a scene/location, and a caption) to include more objects and scene labels, or even captions generated by different systems such as the conceptual captions dataset [Sharma et al. 2018]. Our preliminary experiment with the conceptual captions (CC) dataset shows promising results as this dataset consists of almost three million clean captions. CC is quite unlike curated vision-and-language datasets,

which is ideal to leverage this additional data for a diverse range of vision and language tasks like ours.

Word sense disambiguation. Our word-relatedness approach could be used to tackle similar problems where a word must be selected from among a set of candidates to match a context: That could include lexical selection in machine translation, or word sense disambiguation [Lala and Specia 2018]. We believe our approach could also be useful in multimodal machine translation, where an image caption must be translated using not only the text but also the image content [Barrault et al. 2018].

Deeper architecture for this task. It would be useful to explore further CNN architecture for the task addressed in this thesis. Conneau et al. (2017b) show that training a deep model will result in a major gain in text classification. The Transformer with self attention mechanism Vaswani et al. (2017) also takes advantage of deeper models to learn robust features. As we demonstrate in Chapter 5 the correlation between short text could not be learned with a shallow Transformer. Therefore, there is potential for learning text relations with deeper model like the Universal Sentence Encoder [Cer et al. 2018].

Learning image content and natural language jointly via contextual embedding. In Chapter 5 we showed the power of the pretrain-then-transfer learning approach with BERT [Devlin et al. 2019], which outperforms our approach in this task with a simple tuning. Lu et al. (2019) present Vision and Language BERT (ViLBERT), a joint model for text and image that pre-trained on a large dataset, such as conceptual captions, for visual grounding. This approach was inspired by [Sun et al. 2019] who leveraged a video sequence into BERT. Most recent state-of-the-art approaches are moving in this direction: Tan and Bansal (2019) use a cross-modality model with a more specific design for in-domain datasets (*i.e.* COCO [Lin et al. 2014]) for pre-training, Zhou et al. (2019) show promising results in this research direction of joint language-vision pretraining with universal image-text representation learning, however, there is much research to be done to get to the point of text spotting trained jointly with a visual context system.

Finally, it is my hope that the framework presented in this thesis will lay the foundation for new and exciting research that combines modern vision problems such as OCR with pure NLP techniques in a practical way.

Bibliography

P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

X. Bai, M. Yang, P. Lyu, Y. Xu, and J. Luo. Integrating scene text and visual appearance for fine-grained image classification. *IEEE Access*, 6:66322–66335, 2018.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

L. Barrault, F. Bougares, L. Specia, C. Lala, D. Elliott, and S. Frank. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 304–323, 2018.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. Photoocr: Reading text in uncontrolled conditions. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 2003.

S. Blok, D. Medin, and D. Osherson. Probability from similarity. In *AAAI Spring Symposium on Logical Formalization of Commonsense Reasoning*, 2003.

S. V. Blok, D. L. Medin, and D. N. Osherson. Induction as conditional probability judgment. *Memory & Cognition*, 35(6):1353–1364, 2007.

A. Bordes, J. Weston, and N. Usunier. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 165–180. Springer, 2014.

A. Budanitsky and G. Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and other lexical resources*, volume 2, pages 2–2, 2001.

A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

J. Camacho, L. Espinosa-Anke, and S. Schockaert. Relational word embeddings. *arXiv preprint arXiv:1906.01373*, 2019.

J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64, 2016.

D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1, 2007.

M. Chen, Q. Tang, S. Wiseman, and K. Gimpel. A multi-task approach for disentangling syntax and semantics in sentence representations. *arXiv preprint arXiv:1904.01173*, 2019.

A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2, 2016.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017a.

A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, 2017b.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

T. Dozat. Incorporating nesterov momentum into adam. 2016.

S. Fang, H. Xie, Z.-J. Zha, N. Sun, J. Tan, and Y. Zhang. Attention and language ensemble for scene text recognition with convolutional sequence modeling. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 248–256. ACM, 2018.

Y. Gao, Y. Chen, J. Wang, and H. Lu. Reading scene text with attention convolutional sequence modeling. *arXiv preprint arXiv:1709.04303*, 2017.

S. K. Ghosh, E. Valveny, and A. D. Bagdanov. Visual attention models for scene text recognition. *arXiv preprint arXiv:1706.01487*, 2017.

R. Gomez, B. Shi, L. Gomez, L. Numann, A. Veit, J. Matas, S. Belongie, and D. Karatzas. Icdar2017 robust reading challenge on coco-text. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1435–1443. IEEE, 2017.

A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The journal of machine learning research*, 13(1):307–361, 2012.

H. He, K. Gimpel, and J. Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1576–1586, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

G. E. Hinton. Products of experts. 1999.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8), 1997.

I. Iacobacci. Lstmembed: Learning word and sense representations from a large semantically annotated corpus with long short-term memories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1685–1695, 2019.

I. Iacobacci, M. T. Pilehvar, and R. Navigli. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105, 2015.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

M. Jaderberg. *Deep learning for text spotting*. PhD thesis, University of Oxford.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Deep structured output learning for unconstrained text recognition. *arXiv preprint arXiv:1412.5903*, 2014a.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014b.

M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep features for text spotting. In *ECCV (4)*, 2014c.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 2016.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, 2017.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

C. Kang, G. Kim, and S. I. Yoo. Detection and recognition of text embedded in online images via neural context models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

S. Karaoglu, R. Tao, T. Gevers, and A. W. Smeulders. Words matter: Scene text for image classification and retrieval. *IEEE Transactions on Multimedia*, 19(5):1063–1076, 2017a.

S. Karaoglu, R. Tao, J. C. van Gemert, and T. Gevers. Con-text: Text detection for fine-grained object classification. *IEEE transactions on image processing*, 26(8): 3965–3980, 2017b.

D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. de las Heras. Icdar 2013 robust reading competition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013.

D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, et al. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160. IEEE, 2015.

A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 1992.

M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966, 2015.

C. Lala and L. Specia. Multimodal lexical translation. In *proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

Y. Levine, B. Lenz, O. Dagan, D. Padnos, O. Sharir, S. Shalev-Shwartz, A. Shashua, and Y. Shoham. Sensebert: Driving some sense into bert. *arXiv preprint arXiv:1908.05646*, 2019.

M. Liao, P. Lyu, M. He, C. Yao, W. Wu, and X. Bai. Mask textspotter: An end-to-end trainable neural network for spotting text with arbitrary shapes. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 2014.

P. Lison and J. Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. 2016.

P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. Icdar 2003 robust reading competitions. In *ICDAR*, volume 2003. Citeseer, 2003.

L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

M. Mancini, J. Camacho-Collados, I. Iacobacci, and R. Navigli. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111, 2017.

M. Merler, C. Galleguillos, and S. Belongie. Recognizing groceries in situ using in vitro training data. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

Y. Miao, M. Gowayyed, and F. Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015.

R. Mihalcea, C. Corley, C. Strapparava, et al. Corpus-based and knowledge-based measures of text semantic similarity. In *Aaai*, volume 6, pages 775–780, 2006.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 2013b.

G. A. Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

A. Mishra, K. Alahari, and C. Jawahar. Top-down and bottom-up cues for scene text recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

J. Mueller and A. Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *thirtieth AAAI conference on artificial intelligence*, 2016.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

R. Navigli and S. P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

S. Nazma, D. Kamesh, J. Sastry, and S. Venkateswarlu. Camera based text to speech conversion, obstacle and currency detection for blind persons. *Indian Journal of Science and Technology*, 9(30), 2016.

L. Neumann and J. Matas. A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision*. Springer, 2010.

A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.

Y. Patel, L. Gomez, M. Rusinol, and D. Karatzas. Dynamic lexicon generation for natural scene images. In *European Conference on Computer Vision*, pages 395–410. Springer, 2016.

J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

M. T. Pilehvar and N. Collier. De-conflated semantic representations. *arXiv preprint arXiv:1608.01961*, 2016.

S. Prasad and A. Wai Kin Kong. Using object information for spotting text. In *The European Conference on Computer Vision (ECCV)*, September 2018.

S. Priambada and D. H. Widyantoro. Levensthein distance as a post-process to improve the performance of ocr in written road signs. In *2017 Second International Conference on Informatics and Computing (ICIC)*. IEEE, 2017.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

C. Raffel and D. P. Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, 2015.

N. Rajkumar, M. Anand, and N. Barathiraja. Portable camera-based product label reading for blind people. *International Journal of Engineering Trends and Technology (IJETT)–Volume*, 10, 2014.

P. Rajpurkar, R. Jia, and P. Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

A. Sabir, F. Moreno-Noguer, and L. Padró. Enhancing text spotting with a language model and visual context information. In *CCIA*, pages 271–280, 2018a.

A. Sabir, F. Moreno-Noguer, and L. Padró. Visual re-ranking with natural language understanding for text spotting. In *Asian Conference on Computer Vision*, pages 68–82. Springer, 2018b.

A. Sabir, F. Moreno, and L. Padró. Semantic relatedness based re-ranker for text spotting. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

A. Sabir, F. Moreno-Noguer, and L. Padró. Textual visual semantic dataset for text spotting. *arXiv preprint arXiv:2004.10349*, 2020.

D. Sánchez, M. Batet, D. Isern, and A. Valls. Ontology-based semantic similarity: A new feature-based approach. *Expert systems with applications*, 39(9):7718–7728, 2012.

A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 2015.

P. Sharma, N. Ding, S. Goodman, and R. Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

M. Shilman and P. Viola. Spatial recognition and grouping of text and graphics. In *Proceedings of the First Eurographics conference on Sketch-Based Interfaces and Modeling*. Eurographics Association, 2004.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

C. Sun, A. Myers, C. Vondrick, K. Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7464–7473, 2019.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5103–5114, 2019.

M. Tan, C. Dos Santos, B. Xiang, and B. Zhou. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473, 2016.

P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *European conference on machine learning*, pages 491–502. Springer, 2001.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

K. Wang and S. Belongie. Word spotting in the wild. *Computer Vision–ECCV 2010*, 2010.

K. Wang, B. Babenko, and S. Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.

T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012.

X. Wang, W. Jiang, and Z. Luo. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016a.

Z. Wang, H. Mi, and A. Ittycheriah. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, 2016b.

Z. Wang, W. Hamza, and R. Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

L. Xing, Z. Tian, W. Huang, and M. R. Scott. Convolutional character networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9126–9136, 2019.

K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, 2015.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene text recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4042–4049, 2014.

W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256. Association for Computational Linguistics, 2011.

W. Yin and H. Schütze. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*, 2016.

W. Yin, H. Schütze, B. Xiang, and B. Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.

P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

C. Zhou, C. Sun, Z. Liu, and F. Lau. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*, 2015.

L. Zhou, H. Palangi, L. Zhang, H. Hu, J. J. Corso, and J. Gao. Unified vision-language pre-training for image captioning and vqa. *arXiv preprint arXiv:1909.11059*, 2019.

A. Zhu, R. Gao, and S. Uchida. Could scene context be beneficial for scene text detection? *Pattern Recognition*, 2016.

G. Zhu and C. A. Iglesias. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, 2016.