Research Master 2 SIC

Internship report

Patch-based identification of lexico-semantic relations

Realized by

Nesrine Bannour

ENSEA / University of Cergy-Pontoise / CNRS UMR 8051
6 avenue du Ponceau, 95014 Cergy-Pontoise Cedex, France

The date of presentation : 19 September 2019

Jury :

Gaël Dias	Full Professor at the University of Caen Normandie	Supervisor
Youssef Chahir	Associate Professor at the University of Caen Normandie	Supervisor
Dan Vodislav	Full Professor at the University of Cergy-Pontoise	President
Ngoc Son Vu	Associate Professor at the University of Cergy-Pontoise	Reviewer

Université / / Paris Seine

Acknowledgments

On completion of this internship work, I grab the sacred opportunity to express my heartfelt gratitude to all those who contributed in making my training time in GREYC Laboratory and Crédit Agricole Brie Picardie a worthwhile experience that lived up to my high expectations.

First and foremost, I owe a tremendous debt of thanks to my supervisors Gaël DIAS and Youssef CHAHIR for their unwavering assistance and utmost support in ensuring the progression of the internship scheme. I really want to thank them for giving me from their precious time, for nurturing my sustained interest in the internship subject and for providing me with the key documents that helped me to write my report in a comprehensive manner.

Special appreciations go to Houssam AKHMOUCH and Claire FELDMAN for their assistance and guidance in Crédit Agricole Brie Picardie. I deeply recognize the considerable cooperation and the extreme sympathy of all the persons I got the chance to know in the GREYC Laboratory as well in Crédit Agricole Brie Picardie.

Last but not least, I would like to express my gratitude to my university supervisor, Ngoc Son VU, for his helpful direction and assistance in writing this report.

Finally, I cannot express enough thanks to my family and friends for their support and encouragement throughout this internship.

Abstract: Identifying whether a lexico-semantic relation holds between a pair of words or not is essential for many NLP and Information Retrieval applications. Therefore, multiple strategies have been proposed but most of them focus on identifying a single relation. However, a lot of recent studies show that multi-task strategies can considerably improve the classification process of multiple lexico-semantic relations at the same time. In our work, we consider existing strategies (i.e. binary and multi-task classification) but with a novel input representation of a given word pair. Indeed, we propose to investigate the introduction of related words in terms of cosine similarity measure (so called neighbors) as the new input feature in order to get a more generic idea of the relation that may exist between a pair of words. As such, the concatenation of the continuous distributional representations of the words in a pair and their neighbors forms the patch representation. Evaluation results over a set of gold-standard datasets (RUMEN, ROOT9, Weeds, Bless) show that consistent and significant improvements can be obtained by including these input features, mainly when attention mechanisms are applied based on the PageRank algorithm.

Keywords: Lexico-semantic relations, Binary classification, Multi-task learning, Neural networks, Attention mechanisms, PageRank, Asymmetric relations, Symmetric relations.

Contents

Chapter 1

Introduction

Recognizing the exact nature of the semantic relation holding between a given pair of words is crucial for many Natural Language Processing (NLP) applications such as question answering [Dong 2017], query expansion [Kathuria 2017] or text summarization [Gambhir 2016]. A variety of symmetric and asymmetric lexico-semantic relations among words can be numerated such as synonymy (e.g. phone $\longleftrightarrow$ telephone), co-hyponymy (e.g. phone $\longleftrightarrow$ monitor), hypernymy (e.g. phone $\rightarrow$ speakerphone) or meronymy (e.g. phone $\rightarrow$ mouthpiece), etc.

Numerous approaches have been proposed to identify one particular semantic relation of interest (i.e binary classification) [Snow 2005, Roller 2014, Weeds 2014, Shwartz 2016b, Nguyen 2017, Glavas 2017]. Many studies focus on hypernymy, which lies at the core of human cognition and enables generalization [Wang 2018]. However, considerable research effort exist on antonomy [Nguyen 2017], meronymy [Glavas 2017] and co-hyponymy [Weeds 2014]. On the other hand, several methods have been proposed to discriminate between multiple semantic relations that hold between a pair of words [Guggilla 2016, Shwartz 2016a]. Some works [Liu 2017, Santus 2016] showed that learning features that encode one lexical relation can improve the ability to identify another lexical relation (i.e co-hyponymy and hypernymy [Weeds 2014]). But, this remains a difficult task since it's been hard to distinguish between certain relations (e.g synonymy and hyperonymy) [Shwartz 2016a]. Within this scope, multi-task strategies [Attia 2016, Balikas 2019] have been proposed, which consist of a set of binary classifiers jointly learned. Each classifier is trained to learn whether a specific lexico-semantic relation holds between a pair of given words. The idea behind using multi-task strategies is to enhance the classification performance if the tasks are correlated.

All these studies in the literature take as inputs the distributional and/or path-

based representations of the pair of words to identify the lexico-semantic relation. In our work, we would like to consider not only the representation of the given pair of words but also the neighbors of these words in terms of cosine similarity in the embedding space. The underlying idea of using the nearest neighbors of the pair of words is an analogy with the patch definition in image processing. The patch is defined as a small portion of an image and a container of pixels sharing similar properties [Alkinani 2017].

Once our notion of patch is defined, we also explore the similarity between patches as an analogy to the similarity between words (e.g. cosine similarity).

Recently, attention mechanisms have become effective to obtain higher results in many domains like image processing [Mnih 2014] and also Natural Language Processing [Kim 2017, Lin 2017, Eriguchi 2016, Bahdanau 2014, Rocktäschel 2015, Rush 2015]. Therefore, we include two attention mechanisms in our input representation to better guide our model to distinguish the valuable information from our large input vectors.

In this report, we introduce our strategy to identify lexico-semantic relations that hold between pairs of words by focusing on:

- The continuous distributional representation of word pairs and their neighbors using word embeddings (here GloVe),
- The pattern-based encoding of word pairs and their neighbors using BiLSTM,
- The definition of binary and multi-task classification strategies,
- The definition of attention mechanisms grounded on PageRank and,
- The evaluation over four gold-standard datasets (RUMEN, ROOT9, Bless and Weeds).

Our results show that including the distributional representation of k similar words of a given pair leads to significant improvements, in particular when the attention mechanisms are applied, whatever the different architectures.

Chapter 2

Related Work

Contents

In the following, we present different word representations proposed in the literature and review two main approaches for the identification of lexico-semantic relations, namely pattern-based models and distributional models. We also report on the major learning strategies used in our study and introduce the concepts of patch and attention mechanisms.

2.1 Word embeddings

Word embeddings are vector representations of word meanings, which are dense, distributed, fixed-length, built using word co-occurrence statistics as per the distributional hypothesis This hypothesis was suggested by [Harris 1954] and it consists of the assumption that words with similar contexts have the same meaning. Word embeddings can be categorized into two types, depending on the models used to induce them. The first category is prediction-based models that use local data (e.g. a word's context) and the second one is count-based models that take into consideration global information such as word counts and frequencies [Almeida 2019].

2.1.1 Prediction-based models

The progress of prediction-based is strongly linked with the progress of neural language models. In fact, many studies show that neural language models were delivering better results at modelling language [Bengio 2003a, Bengio 2003b, Mnih 2007, Collobert 2008]. The main contributions of word embeddings comes with the models introduced by [Mikolov 2009, Mikolov 2010]. In particular, [Mikolov 2010] uses recurrent neural networks claiming that the model would remember long contexts without the need to specify explicitly how many words to consider as context. Later on, two models for learning embeddings were proposed [Mikolov 2013a, Mikolov 2013b]: namely the continuous bag-of-words (CBOW) and skip-gram (SG) models. The CBOW model predicts a center word based on its given context words, while the SG model predicts the probabilities of a word being a context word for a given target word. The main objective of these unsupervised feed-forward neural networks is to improve their predictive ability. Variants of these models were proposed by [Mikolov 2013b] to optimize and speed up the training process by using negative sampling. Word2Vec[1] toolkit provides an efficient implementation of these two architectures. Recent contributions propose other improvements of SG models and a new toolkit is made available by Facebook, namely fastText[2] [Joulin 2016]. In particular, good performance was achieved with the fastText model for word representations, which is making use of character-level representations.

2.1.2 Count-based models

The count-based models are unsupervised methods that learn embeddings by constructing a co-occurrence matrix that counts how frequently each word appears in a context in a large corpus [Turney 2010]. Latent Semantic Analysis [Deerwester 1990] is the earliest example of generating word embeddings by applying singular value decomposition on word-context matrices. Multiple count-based models were proposed later on to induce word embeddings [Collobert 2008, Mnih 2008]. [Lebret 2013] reported better results by applying a Hellinger Principal Component Analysis transformation on the word-context matrices. Recently, the main contribution for count-based models is the Global Vector for Word Representation (GloVe)

[1] https://code.google.com/archive/p/word2vec/
[2] https://research.fb.com/projects/fasttext/

model[3] proposed by [Pennington 2014]. This model learns word embeddings by encoding how often two words appear within a given window, i.e. if two words co-occur multiple times, it means that they have a semantic similarity. Authors proved that this model delivers better results than all previous count-based models as well as prediction-based models.

2.1.3 Context-dependent models

In 2018, two successful context-dependent models were proposed. These two models go beyond traditional embeddings that are Word2Vec, fastText or GloVe. They are based on the assumption that a good model should understand the different meanings of words given the surrounding text. The first model is Embeddings from Language Models (ELMo) proposed by [Peters 2018] that uses deep learning models to induce contextualized word representation. Instead of having fixed vectors for the words, ELMo creates vectors by passing text through the deep learning model, i.e. a word can have different embeddings depending on its context and its position in a sentence. The second model is Bidirectional Encoder Representations from Transformers (BERT) proposed by [Devlin 2018]. This model provides as well a context-sensitive embedding by making use of transformers, an attention mechanism that learns contextual relations between words. It has been shown that BERT produces excellent word embeddings, achieving state-of-art results on various NLP tasks.

In our study, we will use the pre-trained GloVe embeddings as we are focusing on single individual words. As such, there is no need for context-dependent models. Moreover, note that any embedding model could be used as our objective is to evaluate how much neighbors can help in lexico-semantic relation identification. As such, we are not especially interested in overall performance. Nevertheless, we are aware that further work will have to be endeavored to verify if the models developed in this work can also adapt to context-dependent embeddings. Indeed, lexico-syntactic patterns between words could play the role of textual context.

[3] https://nlp.stanford.edu/projects/glove/

2.2 Semantic relation classification strategies

Two main approaches are being used to classify a pair of words into a fixed lexico-semantic relation, or to classify it as random if the words are unrelated: pattern-based approaches and distributional approaches. In this section, we review these two approaches and the main learning strategies discussed in the literature.

2.2.1 Pattern-based approaches

Pattern-based methods rely on the lexico-syntactic patterns, which connect two words (e.g. X *such as* Y). The pioneering work is proposed by [Hearst 1992], who has found out that defining specific patterns implies accurate results in identifying semantic relations such as hypernymy. Since then, several variations of pattern-based methods have been proposed to detect hypernymy, in particular [Snow 2005, Ritter 2009, Roller 2018]. Patterns may be predefined, or learned automatically [Kozareva 2010, Snow 2005, Snow 2006]. In this situation, words are represented by a set of patterns and the relevant ones are those who get higher learning weights by the classifier. [Snow 2005] showed that learned patterns incorporate those manually defined by [Hearst 1992]. However, a common problem of lexico-syntactic patterns is their sparsity: no relation can be detected if the words do not co-occur in exactly the right configuration [Roller 2018]. To overcome the sparsity issue, the focus has recently shifted to representing patterns as continuous vectors using neural networks such as BiLSTM [Nguyen 2017, Shwartz 2016b]. In fact, demonstrated good performance proved the generalization efficiency of these models for single semantic relation classification ([Shwartz 2016b] for hypernymy and [Nguyen 2017] for antonymy). In our work, we propose to include these continuous pattern representations as learning features and we aim to represent all patterns independently of the tackled semantic relation.

2.2.2 Distributional approaches

Distributional methods represent another category to capture the semantics between a pair of words and to alleviate the sparsity problem in pattern-based models. It consists of characterizing the semantic relation between two words based on their distributional representations, i.e. vectors based on their distribution separately across large corpora [Wang 2018]. In this case, the words are represented by the concate-

nation of their vectors ([Baroni 2012, Roller 2014, Weeds 2014] for discrete case and [Shwartz 2016b] for the continuous case) or by their difference ([Weeds 2004] for the discrete case and [Fu 2015, Vylomova 2015] for the continuous case). Distributional methods offer rich representations of lexical meaning but the main drawback of the distributional hypothesis is that it conflates different semantic relations between words. Therefore, specialized similarity measures are required to distinguish different relations [Roller 2018].

Another popular solution to this drawback is to specialize word embeddings for particular relations using external knowledge. In this case, the pre-trained distributional space is post-processed to better reflect properties of a certain relation (e.g. [Wieting 2015, Mrksic 2017] for synonymy and [Vulic 2017] for hypernymy). However, these methods are one-relation specific and could not differentiate between multiple semantic relations at a time. [Nguyen 2017, Shwartz 2016b] show that combining pattern-based approaches and distributional approaches enhance classification performance in latent spaces.

For that purpose, we consider in our models both pattern-based and distributional representations.

2.2.3 Learning strategies

Different learning strategies were proposed to identify the semantic relation that holds between words: binary classification [Snow 2005, Roller 2014, Weeds 2014, Shwartz 2016b, Nguyen 2017, Glavas 2017], multi-class classification [Guggilla 2016, Shwartz 2016a] and multi-task strategies [Attia 2016, Balikas 2019]. In binary classification models, the objective is to detect a single relation at a time. [Snow 2005] build a classifier that learns to decide whether the relation holding between a given pair of nouns, if any, is a hypernymy relation. [Weeds 2014] consider supervised approaches by constructing linear Support Vector Machines and k-Nearest Neighbors classifiers and reported that these models could outperform state-of-the-art unsupervised methods when it comes to distinguishing hypernyms and co-hyponyms. [Shwartz 2016b] tackled as well the hypernymy relation by proposing an improved path-based algorithm using a recurrent neural network (LSTM). [Glavas 2017] focuses on detecting asymmetric lexico-semantic relations: hypernymy and meronomy. A neural architecture, named Dual Tensor model is proposed that models better asymmetry and that requires only distributional vectors of words as input. This

model was evaluated on hypernymy classification and meronomy classification.

Within the multi-class classification strategies, [Guggilla 2016] proposed a lexical semantic relation detection system using convolutional neural networks as part of the CogALex-V semantic shared task[4]. The task is divided into two subtasks: the subtask of detecting if a relation exists between two given words and the subtask of identifying the type of the relation (hypernymy, meronymy, synonymy or antonymy). Within this scope, [Shwartz 2016a] achieved a good performance making it between the top-ranked methods in this multi-class problem.

[Attia 2016] introduced a multi-task strategy by proposing a multi-task convolutional neural network to jointly learn the semantic relation between words within the second subtask of the CogAlex-V semantic shared task. Another multi-task strategy has been proposed by [Balikas 2019] to concurrently learning semantic relations with the assumption that the learning process of a given semantic relation may be improved by jointly learning another semantic relation. These two studies encode word pairs based on their distributional approach and do not take into consideration the pattern-based continuous representations.

In our work, besides binary architectures, we propose multi-task architectures for symmetric and asymmetric semantic relations.

2.3 Patch and attention mechanisms

In the last several years, there have been advances in both the field of Computer Vision and NLP. The emergence of deep learning techniques has improved many applications in image processing. This rise and success of convolutional neural networks and recurrent neural networks in particular, has led to multiple studies in many other fields. For instance, in the field of NLP significant results have been achieved (e.g [Kim 2014] for sequence classification, [Zeng 2014] for relation classification). Our work is inspired by the Computer Vision field, in particular the patch definition in image processing. A patch is defined in image processing as a small portion of an image and a container of pixels sharing similar properties. Using patches instead of processing one pixel at a time of the image is preferable for many image processing applications such as image denoising [Alkinani 2017]. Since many NLP approaches consider words as basic units, we would like to consider a similar

[4]https://sites.google.com/site/cogalex2016/home/shared-task

definition of the patch in NLP. In fact, instead of modeling only the representation of the given pair of words in our models, we want to introduce the representation of the neighbor words of the given pair in terms of cosine similarity measure in the embedding space. As a consequence, a word is not represented by its unique lexical unit but also by all its neighboring lexical units. Note that a very recent work [Jiménez 2019] proposes a similar idea but in the discrete case with set-based metrics.

Our initial findings showed that the direct introduction of neighbor words does not lead to improvements. This is mainly due to the vanishing of the focus word when introducing neighbors. Indeed, in this case, all words become equal, although some importance should be given to the main concept of the set of words. For that purpose, we propose to implement an attention mechanism. Recently, attention mechanisms have become effective to obtain higher results in Computer Vision [Mnih 2014] and Natural Language processing [Kim 2017, Lin 2017, Eriguchi 2016, Bahdanau 2014, Rocktäschel 2015, Rush 2015]. Within our work, we define an attention mechanism based on the PageRank algorithm [Brin 1998] that aims to give a weight of centrality to each of the neighbors when considering the graph of all neighbors. As such, the more a word is central in the graph, the more it will receive a high score. These scores will then be used as attention weights to the corresponding continuous representations of the neighbors. The objective of this mechanism is to improve the predictive ability of our system based on the importance score of each word embedding in the patch.

Chapter 3

Baseline Model

Contents

[Balikas 2019] proposed a fully-shared multi-task architecture to identify lexico-semantic relations between words by considering the concatenation of the distributional representation of words as input features. In this chapter, we briefly present their new work[1], as part of the PhD. thesis of Houssam AKHMOUCH at the CNRS research laboratory GREYC[2] in collaboration with Crédit Agricole Brie Picardie[3], as a baseline model to our contribution. We start by defining the word pair continuous representations and then we describe the neural network architectures proposed as well as some experimental setups.

3.1 Word pair continuous representations

3.1.1 Distributional representation

To represent a word pair, three different representations are proposed: vector concatenation, vector difference and cosine similarity. Let (x, y) be a word pair and

[1] This work is under revision for a journal issue.
[2] https://www.greyc.fr/
[3] https://www.ca-briepicardie.fr

(w_1, w_2) their distributional representations (here GloVe embeddings of dimension $d = 300$). Let's note the concatenation of the distributional vectors $w_1 \oplus w_2$, $w_1 \ominus w_2$ the vector difference and $cos(w_1, w_2)$ the cosine similarity measure defined in equation 3.1. These three representations are concatenated and introduced as input features, i.e. $(w_1 \oplus w_2$,$\mathrm{w}_1 \ominus w_2, cos(w_1, w_2))$.

$$cos(w_1, w_2) = \frac{\sum_{i=1}^{d} w_1^i \times w_2^i}{\sqrt{\sum_{i=1}^{d} {w_1^i}^2} \times \sqrt{\sum_{i=1}^{d} {w_2^i}^2}} \tag{3.1}$$

3.1.2 Pattern-based representation

Along with the distributional representations, a word pair is also represented with the continuous vectors encoding the patterns. Patterns are defined as word sequences that occur between the pair of words (x, y) within a span of 10 (chosen empirically) words maximum of the same sentence. Table 3.1 shows some examples of patterns between two words.

Relation	Pattern
Synonymy	**error** or **fault**
Hypernymy	**unit** that includes **screen**
Co-hyponymy	**pineapple** and **apricot**
Meronymy	**bowl** from the world of **glass**
Random	**reference** in the book of **mormon**

Table 3.1: Examples of patterns of a word pair (in bold).

While some patterns can easily indicate the semantic relation, other sequences can give wrong link predictions. For this matter, AKHMOUCH et al., decided to keep only the two most frequent patterns for each pair of words. To encode these patterns, a Bidirectional Long-short Term Memory (BiLSTM) neural network has been proposed and zero padding has been used to ensure that every sequence has the same length. These two most frequent patterns are then averaged to obtain a single representation, noted $h_*^{w_1,w_2}$. As such the overall input of the model is noted:

$$X = (w_1 \oplus w_2, w_1 \ominus w_2, cos(w_1, w_2), h_*^{w_1,w_2}). \tag{3.2}$$

3.2 Multi-task architectures

Based on the assumption that a shared layer is efficient when concurrently learning multiple tasks [Balikas 2019], the overall architecture presented in figure 3.1 consists of a shared part as well as several private parts. The layer S is the shared one encoding the common information between tasks as defined in Equation 3.3.

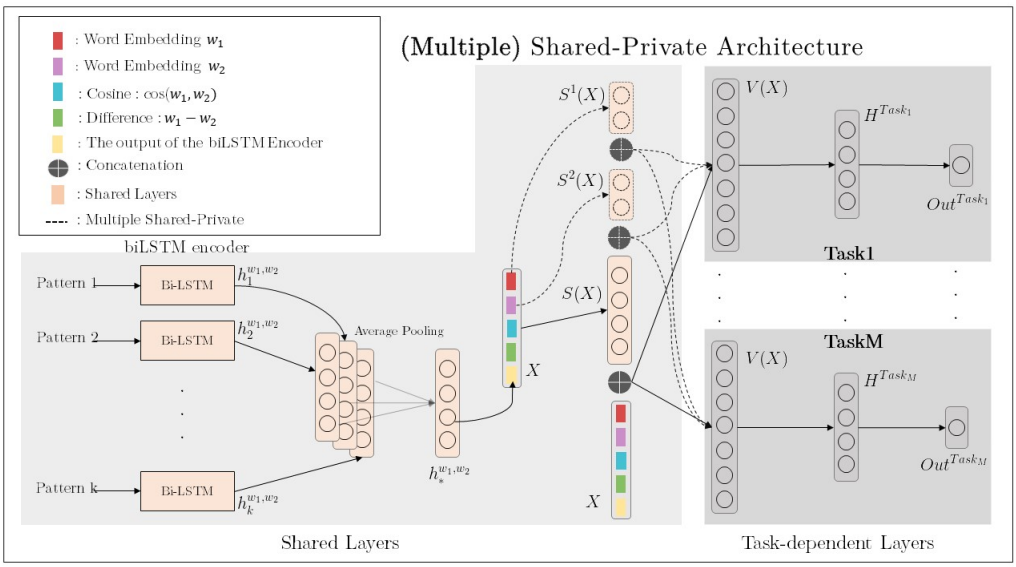

Figure 3.1: The generic (Multiple) Shared-Private Architecture. Different configurations of $V(X)$ allow to propose fully-shared, single shared-private, multiple shared-private architectures for M tasks.

$$S(X) = \sigma(W_S X + b_S) \tag{3.3}$$

Generic information about words is learned in this shared layer. However, this may be seen as a limitation for asymmetric relations (hypernymy or meronymy) where the word order in the input pair matters. To address this issue, two more extra shared layers are proposed, one for each input word (S^1 for w_1 and S^2 for w_2) defined in Equation 3.4.

$$S^i(X) = \sigma(W_S^i w_i + b_S^i), \quad \forall i, \quad 1 \leq i \leq 2 \tag{3.4}$$

As a result, these extra shared layers encode useful information that individual words may contain alone to decide whether two words are semantically related.

$S(X)$, $S^1(X)$ and $S^2(X)$ are shared across all tasks. However, in [Liu 2017], it has be proven that the classification performance is enhanced if all concurrent tasks receive both shared and private representations. For this matter, shared-private architectures are introduced where the input vector is represented with the three shared representations $S(X)$, $S^1(X)$ and $S^2(X)$, plus the initial input representation X fed to two task-dependent layers, namely H^{T_j} and Out^{T_j} respectively defined in Equations 3.5 and 3.6 for a specific task T_j .

$$H^{T_j} = \sigma(W_H^{T_j} V(X) + b_H^{T_j}) \tag{3.5}$$

$$Out^{T_j} = \sigma(W_O^{T_j} H^{T_j} + b_O^{T_j}) \tag{3.6}$$

The neural network model parameters are updated by minimizing the binary cross-entropy function $E(T_j)$ defined in Equation 3.7. Mini batch training is used with b being the size of the batch. The loss function takes as input Out^{T_j} and y^{T_j}, where $y^{T_j} = 1$ if x and y are T_j-related and $y^{T_j} = 0$ otherwise.

$$E(T_j) = -\frac{1}{b}\sum_{i=1}^{b} y_i^{T_j}.\log(Out_i^{T_j}) + (1 - y_i^{T_j}).\log(1 - Out_i^{T_j}) \tag{3.7}$$

Concretely, the shared parameters of the shared layers (such as W_S and W_S^i in equations 3.3 and 3.4) are updated by minimizing $E(T_j)$ alternatively with the other tasks. Whereas the private parameters of the private layers (such as $W_H^{T_j}$ and $W_O^{T_j}$ in equations 3.5 and 3.6) are only updated by minimizing $E(T_j)$ for their specific learning examples. These training steps are also detailed in Algorithm 1.

From this generic model, three different specific architectures are generated. First, the **fully-shared** model originally proposed by [Balikas 2019]. In this architecture, $V(X) = S(X)$, meaning that all the input information is shared and it's the only information used for classification. Second, the **single shared-private** model with both shared and private representations but without individual word information. In fact, $V(X) = (S(X), X)$. The third architecture proposed is the **multiple shared-private** model. In this case, $V(X) = (S(X), S^1(X), S^2(X), X)$. So, both shared and private representations are used for the decision process along with the dedicated parts for the embeddings of the input words. These parts are introduced to improve the performance on asymmetric relations.

Algorithm 1 The Multiple shared-private Training Process.

Data: Word pairs for each of the M tasks, batch size b, iterations

it = 1 ;

while *it* < *iterations* **do**

  for $i = 0;\ i < M;\ i = i + 1$ **do**

    Randomly select a batch of size b for T_i called $batch_i$

    for (w_1, w_2) *in* $batch_i$ **do**

      Compute Out^{T_i}

    end

    Update the parameters according to $E(T_i)$ from $batch_i$

    Calculate performance on the validation set of T_i.

  end

end

Within this scope, another specific architecture has been proposed where patterns are not shared through the network. The underlying idea is that patterns might be relation-dependent and should only be used as private layers. Therefore, the input vector X given in equation 3.2 would be ablated from its $h_*^{w_1,w_2}$ argument, such that $X = (w_1 \oplus w_2, w_1 \ominus w_2, cos(w_1, w_2))$. As a consequence, $h_*^{w_1,w_2}$ would be directly concatenated to the vector $V(X)$, such that $V(X) = (S(X), S^1(X), S^2(X), X, h_*^{w_1,w_2})$.

Note that these architectures have been tested for concurrent learning of two and three tasks. But in our study, we focus on one-class architectures and the concurrent learning of two tasks.

3.3 Experimental setups

In this section, we present the datasets used to evaluate these multi-task architectures. We explain the lexical split and detail the pattern extraction step. Note that we will use the exact same experimental setups in our study.

3.3.1 Datasets

As reported in Chapter 2, many studies exist for the identification of lexico-semantic relations. [Weeds 2004] proposed the first gold-standard dataset, called Weeds in the context of studies about measures of lexical similarity. Following the same objective, [Baroni 2012] introduced the well-known Bless dataset, and [Santus 2016] compiled

the ROOT9 dataset[4] , which contains word pairs randomly extracted from EVALution [Santus 2015], Lenci/Benotto [Benotto 2015] and Bless [Baroni 2012]. Within the context of concurrent identification of lexico-semantic relations, [Balikas 2019] recently introduced the RUMEN dataset[5] to include synonymy. All datasets are summarized with their specific characteristics in Table 3.2.

	Synonym		Hypernym		Co-hyponym		Meronym		Random	
Dataset	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)	#	0 / 1 / 2 / >2 (%)
RUMEN [Balikas 2019]	6326	44/14/8/34	6326	65/12/5/18	-	-	-	-	6326	93/4/1/2
ROOT9 [Santus 2016]	-	-	2447	21/9/6/64	3200	28/14/8/50	-	-	1100	78/9/4/9
Weeds [Weeds 2004]	-	-	1257	40/13/6/41	2083	60/11/5/24	-	-	6326	93/4/1/2
Bless [Baroni 2012]	-	-	1337	57/9/4/30	3565	34/12/7/40	2943	99/0/0/1	6702	97/1/0/2
ROOT9 (bal.)	-	-	1100	20/9/5/65	1100	29/15/8/48	-	-	1100	78/9/4/9
Weeds (bal.)	-	-	1257	40/13/7/40	1257	61/10/5/24	-	-	1257	94/4/1/1

Table 3.2: Details of the RUMEN, ROOT9, Weeds and Bless datasets. 0 / 1 / 2 / >2 stands for the percentage of word pairs having respectively no pattern, 1 pattern, 2 patterns and more than 2 patterns in the Wikipedia dump. ROOT9 (bal.) and Weeds (bal.) stand for a balanced version of the original ROOT9 and Weeds datasets.

In order to avoid the natural imbalance of the datasets, ROOT9 and Weeds have been modified to ensure that all classes have the same number of pairs. However, RUMEN was designed to be balanced, so it was not modified. These datasets are tested to have an overall overview of the bias that can be introduced by an unbalanced set of learning examples and allow more direct comparisons between datasets.

3.3.2 Lexical split

As proposed in [Levy 2015], a lexical split is proposed to remove possible vocabulary intersection between the test and the training/validation sets. In fact, [Levy 2015] points out that lexical memorization could be performed instead of learning lexico-semantic relations between words, when using distributional representations in the context of supervised learning. To avoid this issue, a lexical split is proposed and we follow the exact same method defined by [Balikas 2019].

[4]https://github.com/esantus/ROOT9

[5]https://github.com/Houssam93/MultiTask-Learning-NLP/tree/master

3.3.3 Pattern extraction

In order to extract the patterns that connect the word pairs of the datasets studied, the English Wikipedia dump[6] has been downloaded and a representative corpus of approximately 4Gb has been extracted. All the articles in this corpus have then been splitted into sentences. Note that these sentences were first pre-processed to remove special characters and stop words. Only patterns that do not exceed a maximum length of 10 words[7] in a same sentence have then been considered as valid word sequences.

In our work, we consider the different architectures introduced in this Chapter as a baseline work to compare with and to prove the efficiency of our contribution. In fact, as the code and the datasets of [Balikas 2019] are available for reproducibility, we particularly test our hypotheses on their specific task. The performance of these architectures will be covered in Chapter 5, along with our results.

[6] https://dumps.wikimedia.org/enwiki/20190220/

[7] Value tuned experimentally.

Chapter 4

Patched-based Classification

Contents

In this chapter, we detail our main contributions. We start by giving our definition of the notion of patch, which is the underlying idea of our work. Then, we continue by defining the similarity measure between patches. We present, the several modifications we have made to the baseline architectures explained in Chapter 3 and introduce the attention mechanism we have used to improve our results. Finally, we present learning configurations.

4.1 Patch definition

As already mentioned in Chapter 2, we aim in our work to include the word neighbors representation in the input vector. The word neighbors are defined as the k most similar words to a source word in terms of cosine similarity in the embedding space. In our case, we calculate the cosine similarity between the given word embedding and the other words in the GloVe embedding space. We then select the k[1] most similar vectors to the given word. Table 4.1 illustrates some examples of words with their neighbors, with descending order.

Once we have the neighbors of both words of the given pair, there are two options in the way of introducing these words. The first idea is to consider their

[1] Several values of this parameter have been evaluated: from 1 to 10.

Word	Neighbors
dog	(dogs, **0.7888**), (cat, **0.6816**), (pet, **0.6292**), (puppy, **0.5936**), (hound, **0.5468**), (horse, **0.5369**), (animal, **0.5316**), (cats, **0.5081**), (canine, **0.5038**), (pets, **0.5020**)
animal	(animals, **0.8182**), (human, **0.5878**), (livestock, **0.5623**), (humans, **0.5392**), (dog, **0.5316**), (meat, **0.5278**), (wildlife, **0.5258**), (pet, **0.5168**), (elephant, **0.5043**), (sheep, **0.50386**)
whole	(entire, **0.7742**), (everything, **0.6365**), (really, **0.5975**), (thing, **0.5972**), (rest, **0.5958**), (part, **0.5879**), (basically, **0.5854**), (this, **0.5829**), (just, **0.5792**), (it, **0.5782**)
unit	(units, **0.7417**), (division, **0.5817**), (operations, **0.5754**), (subsidiary, **0.5044**), (company, **0.4778**), (force, **0.4743**), (battalion, **0.4680**), (brigade, **0.4651**), (officer, **0.4645**), (group, **0.4586**)

Table 4.1: Examples of neighbors with the cosine similarity measure to the initial word (in bold).

distributional representation (i.e. their GloVe embeddings) by concatenating them to the initial input. The second idea would be the consideration of only the cosine similarity measures between all the neighbors without explicitly introducing the distributional vectors. In this case, a word-to-word matrix would be computed between all the neighbors and concatenated to the initial distributed representations of the input word pair. This second idea has been tested but evidenced decreased performance, when compared to the baseline model. However, alone, it shows that some information is encoded in such a matrix. This will be the starting point of the idea of the attention mechanism introduced in section 4.4.

As a consequence, we proceed with the first idea, where instead of having just the embeddings of the pair of words, we include as well the embeddings of all k word neighbors. Figure 4.1 illustrates an example of a word pair and its neighbors. In this example, we consider the word pair ***(animal, dog)*** and we fix $k = 2$ to have two neighbors for each word. The relation to be predicted is the hypernymy relation. The neighbors of the word *animal* are {*animals, human*} and the neighbors of the word *dog* are {*dogs, cat*}. We can notice that including the word *human* in the

input vector increases the possibility of recognizing that the initial word *animal* is a generic term: a hypernym in this case. On the other side, the word *cat* is a good indicator that *dog* is a specific term : a hyponym. So, we think that adding the k most similar words may benefit the relation decision.

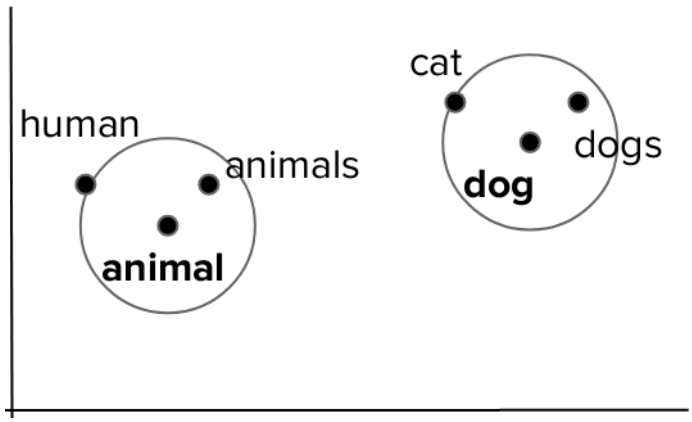

Figure 4.1: Illustrative example of a word pair with its neighbors in the emebdding space.

The underlying idea is to no longer predict the relation that holds between a pair of words but between a pair of patches (concepts), where the meaning may be better estimated.

4.2 Similarity between patches

By analogy with our baseline model, we would like to introduce a similarity measure in our initial input. Therefore, we extend the definition of cosine similarity measure between the given word pair to a similarity measure between patches. This similarity measure between two patches of $(k+1)$ words each is defined in Equation 4.1 and is represented by a concatenated $(k+1) \times (k+1)$ matrix of word-to-word cosine similarity measures, that will be used as input to the learning model. Note that the embeddings of the initial word pair are noted w_{10} and w_{20}, and w_{1i} (resp. w_{2j}) corresponds to the embedding of the ith (resp. jth) neighbor of word 1 (resp. 2).

$$SimPatches(w_1, w_2) = \bigoplus_{i=0}^{k} \bigoplus_{j=0}^{k} cos(w_{1i}, w_{2j}) \tag{4.1}$$

Based on the same assumption that working with patches reflects better the meaning of the words and hence the nature of the lexico-semantic relation, we believe that introducing $SimPatches(.,.)$ enhances the model performance as it indicates the degree of similarity between all pairs of words present in the patches. As follows, we present the $SimPatches(.,.)$ matrix of the previous example, which is introduced as concatenated values in the learning models.

$$SimPatches(animal, dog) = \begin{array}{c} \\ animal \\ animals \\ human \end{array} \begin{array}{c} \begin{array}{ccc} dog & dogs & cat \end{array} \\ \begin{pmatrix} 0.5316 & 0.4692 & 0.2727 \\ 0.5025 & 0.6330 & 0.2738 \\ 0.4414 & 0.4121 & 0.1849 \end{pmatrix} \end{array}$$

4.3 Pattern-based representation of patches

We also extend the pattern definition to adapt it to patches. In the baseline model introduced in Chapter 3, only the two most frequent patterns are kept for each word pair. These patterns are then encoded with a BiLSTM and averaged in the end to get a single representation. In our case, we have many $(k+1)^2$ word pairs. As such, it not conceivable to compute two patterns for each pair of words[2]. In fact, if we choose to include $k = 5$ neighbors, then we will have 36 word pairs. For this matter, we follow the same strategy as the baseline model but by considering only the first frequent pattern for each pair. In this case, we have 36 BiLSTM to encode these patterns and 36 representations that need to be concatenated with the input, instead of just a single one. Following the previous notation, the patterns are represented in Equation 4.2, and consist of the concatenation of $(k+1)^2$ BiLSTM.

$$Patterns(w_1, w_2) = \bigoplus_{i=0}^{k} \bigoplus_{j=0}^{k} h^{w_{1i}, w_{2j}} \tag{4.2}$$

Since this strategy increases the dimension of our input considerably, we propose a second strategy that could be followed. It consists of selecting only the most frequent pattern of all the patterns grouped and get a single representation for all word pairs. This strategy could alleviate the computational costs.

[2]This remains as future work outside the goal of this master thesis.

Finally, a word pair (x,y) is represented by the vector X defined in Equation 4.3, where $\bigoplus_{i=0}^{k} w_{zi}$ represents the concatenation of the word embedding of word w_z plus all the embeddiings of its neighbors.

$$X = (\bigoplus_{i=0}^{k} w_{1i}, \bigoplus_{i=0}^{k} w_{2i}, \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} cos(w_{1i}, w_{2j}), \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} h^{w_{1i},w_{2j}}) \tag{4.3}$$

Note that the neighbor word embeddings are included by descending order, meaning that the most similar vector is concatenated first. This order is important to comply with attention mechanism.

4.4 Attention mechanisms

In our work, we use the same architectures as the baseline model. However, we increase the dimension of our input vector since each word embedding is of dimension 300 and we evaluate different values of k: the number of neighbors to select. Therefore, we find it a necessity to include some attention mechanisms in order to avoid a certain noise that can be added with the neighbor vectors. The overall idea of our first attention strategy, namely $attentionMech_1$, is to give an importance score to each word embedding in the input vector. Therefore, we choose to use the PageRank algorithm [Brin 1998] to get a score for each word in a given patch. As explained previously, the idea is to give more important focus on the central words in the patch. For instance, for the following patch $\{animal, animals, human\}$, one would agree that both words *animal* and *animals* are more central to the overall patch than *human*. As a consequence, we construct an undirected graph from a given patch. Each node in the graph represents a word and each edge is weighted with the cosine similarity measure that links two words in the patch. The PageRank works by counting the number and quality of the links to a word to determine an estimate of how important the word is in its neighborhood. This way, we get an importance vector for each word in a patch. Table 4.2 illustrates the PageRank values two different patches, namely *dog* and *animal*. So, each word embedding is multiplied by its PageRank value as an attention value. Thus, the input embedding vector of each word in the patch is modified with the attention-weighted GloVe embeddings.

Patch	PageRank values
dog	(dog, **0.1724**), (dogs, **0.1513**), (cat, **0.1506**), (pet, **0.1409**), (puppy, **0.1353**), (hound, **0.1258**), (horse, **0.1217**)
animal	(animal, **0.1670**), (animals, **0.1667**), (humans, **0.1434**), (livestock, **0.1341**), (meat, **0.1319**), (human, **0.1311**), (dog, **0.1256**)

Table 4.2: PageRank values for the patches of dog and animal.

We propose a second attention mechanism, namely $attentionMech_2$ that treats the whole set of words (i.e two patches). The overall idea is that specific words play a given role when deciding if two words are in a lexico-semantic relation. While $attentionMech_1$ focuses on the centrality within a given patch, $attentionMech_2$ focuses on the centrality between two patches. Indeed, it is important to acknowledge, which words are central in a set of two patches in order to verify if two words are in a lexico-semantic relation. For instance, between the two patches $\{animal, animals, human\}$ and $\{dog, dogs, cat\}$, one may agree that *animal*, *animals* and *dog*, *dogs* are the central words to make the decision, and *human* and *cat* are subsidiary (but useful) information. As a consequence, we execute the PageRank algorithm on a global graph formed by a pair of patches, where all words are connected and weighted by their cosine similarity. It is important to note that for that purpose, we exclusively use the $SimPatches$ matrix. Thus, the edges within a given patch are not taken into account for the PageRank as they are already computed in $attentionMech_1$. Therefore, we get a single importance vector that includes the scores of each word in the concatenation. Following the same previous example, Table 4.3 illustrates the PageRank values for $k = 6$ and $k = 3$. Through the table 4.3, we can notice that the choice of the number of neighbors to select is crucial. In fact, when we set $k = 6$, the scores don't really reflect the importance of the words (small values). However, when we set $k = 3$, the scores are more representative.

4.5 Binary and multi-task classification

As we already mentioned, we use the same architectures as the baseline model in Chapter 3 : the **fully-shared** architecture, the **single shared-private** architecture,

Initial patches	PageRank values
{**Dog**, **Animal**} ($k = 6$)	(dog, **0.0865**), (dogs, **0.0777**), (dog, **0.0766**), (pet, **0.0741**), (animal, **0.0740**), (animals, **0.0737**), (cat, **0.0716**), (horse, **0.0707**), (meat, **0.0683**), (livestock, **0.0672**), (humans, **0.0667**), (puppy, **0.0658**), (hound, **0.0634**), (human, **0.0634**)
{**Dog**, **Animal**} ($k = 3$)	(animal, **0.1413**), (animals, **0.1398**), (dogs, **0.1320**), (dog, **0.1315**), (pet, **0.1242**), (cat, **0.1172**), (human, **0.1044**), (livestock, **0.1095**)

Table 4.3: PageRank values between two patches.

and the **multiple shared-private architecture** for multi-task architectures. For the binary classification, the baseline architecture is also used where the shared representations are omitted. In this section, we introduce the modifications and changes we made to these architectures.

For the **fully-shared** and **single shared-private** models, no changes are needed, except to the change of the input vector, which turns to be X defined in Equation 4.3. However, for the **multiple shared-private** model, extra shared representations are proposed, one for each input word. Therefore, in our case, we have an extra shared layer for each patch instead of each word (S^1 for patch 1 and S^2 for patch 2). So, the model must be redefined for $S^i(X)$ as in Equation 4.4.

$$S^i(X) = \sigma(W_S^i \bigoplus_{j=0}^{k} w_{ij} + b_S^i), \quad \forall i, \quad 1 \leq i \leq 2 \tag{4.4}$$

After introducing our two attention strategies in the previous section, we have two possibilities to include this attention in our input vector. The first possibility, noted ***seqArchitecture*** consists of applying the two strategies sequentially: the input word embeddings of each patch are weighted with $attentionMech_1$ and then concatenated together to obtain a global representation of $2 \times (k+1)$ vectors. This vector is then weighted with $attentionMech_2$. Note that word ordering is important. Therefore, the final input X vector can be defined as in Equation 4.5.

$$X = (\bigoplus_{i=0}^{k} w_{1i}^{att12}, \bigoplus_{i=0}^{k} w_{2i}^{att12}, \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} cos(w_{1i}, w_{2j}), \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} h^{w_{1i}, w_{2j}}) \tag{4.5}$$

The second possibility, noted ***paralArchitecture***, consists of a parallel process. On the one hand, the input word embeddings of each patch are weighted with $attentionMech_1$ and on the other hand, the concatenation of these input embeddings (non weighted with $attentionMech_1$) is weighted with $attentionMech_2$. So instead of having $2 \times (k+1)$ input vectors weighted twice, we will have $(k+1)$ weighted (with $attentionMech_1$) input vectors for each patch concatenated with $2 \times (k+1)$ weighted $attentionMech_2$ input vectors. We are aware that this architecture increases the dimension of our input vector and some information could be duplicated. However, it is worth testing both strategies as they reveal different cognitive strategies. Therefore, the final input X vector can be defined as in Equation 4.6 for the ***paralArchitecture***.

$$X = (\bigoplus_{i=0}^{k} w_{1i}^{att1}, \bigoplus_{i=0}^{k} w_{2i}^{att1}, \bigoplus_{i=0}^{k} (w_{1i}, w_{2i})^{att2}, \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} cos(w_{1i}, w_{2j}), \bigoplus_{i=0}^{k}\bigoplus_{j=0}^{k} h^{w_{1i}, w_{2j}}) \tag{4.6}$$

4.6 Learning framework

In our work, word embeddings are initialized with the 300-dimensional representations of GloVe [Pennington 2014]. The similar words are extracted using the Gensim python library[3] and the PageRank algorithm is the one implemented in the NetworkX package[4]. For the S, S^1, S^2 and H^{T_j} layers, the number of neurons per layer is respectively 300, 100, 100 and 50. These layers share the sigmoid function as the activation function. As for the learning process, Adam [Kingma 2014] is used as the optimizer with the default parameters of Keras [Chollet 2015]. The network is trained with batches of 64 examples and the number of iterations is optimized to maximize the F_1 score on the validation set. The neural architectures are implemented using Keras with Tensorflow [Abadi 2015] as a back-end.

[3] https://pypi.org/project/gensim/
[4] https://networkx.github.io/

Chapter 5

Results and Evaluation

Contents

In this chapter, we discuss the evaluation results of our models presented in Chapter 4. These models are tested over the four datasets that we described in Chapter 3, namely RUMEN, ROOT9, Weeds and Bless and are compared in performance with the work of [Balikas 2019] and the baseline models introduced in Chapter 3. Results are presented for both binary classification and multi-task architectures. The introduction of the pattern-based representations is discussed in the last section of this chapter.

5.1 Evaluation measures

To evaluate the classification performance of our models, we use the *Accuracy* and the F_1 score measures defined in the following equations where, TP, TN, FP and FN respectively stand for True Positives, True Negatives, False Positives and False Negatives.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

$$\text{F}_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

5.2 Binary classification

5.2.1 RUMEN evaluation

Table 5.1 shows the results of five different experiments for binary classification for the RUMEN dataset. In the **first experiment**, we set the input representation to only the concatenation of the distributional representation of the word pair. In the **second experiment**, we add the cosine similarity measure between the given pair of words. These two experiments are considered as our baselines and the importance of cosine similarity feature is clearly evidenced for both synonymy and hypernymy identification. For the **third experiment**, the patches representation is fed to our models (i.e. the concatenation of the distributional representation of the word pair and its k neighbors). We vary the k parameter, i.e. the number of neighbors to select from 1 to 10. We can notice that adding the distributional representation of the k most similar words decreases the performance of our models. This can be explained by the fact that the dimension of our input increases, when we increment k and since we are dealing with GloVe vectors of dimension 300, this could introduce noise information in the decision process. Moreover, it shows that the simple introduction of neighbors implies a lack of focus, and as a consequence a decrease in performance when compared to the simple concatenation of words in the word pair. In order to evaluate the importance of our $SimPatches(.,.)$ measure, we set our input vector in the **fourth** test to the concatenation of word pair embeddings along with $SimPatches(.,.)$. We can clearly notice that the models outperform all the previous ones for all k values and for both tasks. Compared to the **second** experiment, an improvement of 1.2% in terms of accuracy ($k = 6$) and 1.1% in terms of F_1 score ($k = 6$) is obtained for the synonymy identification and an improvement of 1.1% in terms of accuracy ($k = 9$) and 1.2% in terms of F_1 score ($k = 9$) for

the hypernymy classification. Even though the improvement values are small, we think that $SimPatches(.,.)$ is an important feature with valuable information. To better confirm this statement, in the **fifth** experiment, we concatenate the patches representation and $SimPatches(.,.)$ measure. The results are better than those of the **third** test but do not outperform the **fourth** test. Therefore, we can conclude that the patches representation should be ameliorated and that our patch idea may benefit the classification process.

	Inputs		Synonym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	Concat		0,756	0.760	0.744	0.776	0.776	0.768	0.741	0.714
	Concat + cos		0,817	0,819	0,806	0,833	0,790	0,766	0,741	0,792
	Patches	K = 1	**0,713**	**0,718**	**0,701**	**0,736**	0,747	0,712	0,699	0,726
		K = 2	0,675	0,683	0,664	0,685	0,702	0,662	0,649	0,676
		K = 3	0,655	0,664	0,644	0,685	**0,765**	**0,728**	**0,728**	0,729
		K = 4	0,684	0,697	0,666	0,730	0,757	0,723	0,713	**0,734**
		K = 5	0,660	0,670	0,647	0,694	0,718	0,677	0,670	0,685
		K = 6	0,7	0,705	0,689	0,721	0,723	0,683	0,676	0,691
		K = 7	0,684	0,695	0,668	0,725	0,732	0,685	0,696	0,676
		K = 8	0,684	0,687	0,676	0,697	0,731	0,687	0,691	0,682
		K = 9	0,663	0,678	0,645	0,714	0,723	0,678	0,680	0,677
		K = 10	0,648	0,659	0,635	0,684	0,730	0,687	0,688	0,685
	Concat + SP	K = 1	0,822	0,824	0,813	0,835	0,793	0,768	0,744	0,794
		K = 2	0,822	0,824	0,811	0,837	0,795	0,771	0,747	0,795
		K = 3	0,826	0,827	0,815	0,840	0,795	0,772	0,745	0,801
		K = 4	0,828	0,828	**0,821**	0,835	0,790	0,767	0,737	0,799
		K = 5	0,828	0,829	0,819	0,840	0,792	0,766	0,744	0,790
		K = 6	**0,829**	**0,830**	0,819	0,841	0,792	0,768	0,743	0,795
		K = 7	0,827	0,829	0,817	**0,842**	0,791	0,766	0,741	0,792
		K = 8	0,823	0,824	0,814	0,834	0,794	0,769	0,745	0,795
		K = 9	0,822	0,823	0,817	0,829	**0,801**	**0,778**	**0,753**	**0,805**
		K = 10	0,821	0,822	0,814	0,831	0,798	0,774	0,747	0,803
	Patches + SP	K = 1	0,792	**0,793**	0,786	0,799	0,767	0,735	0,721	0,750
		K = 2	0,779	0,778	0,776	0,781	0,759	0,722	0,721	0,723
		K = 3	0,779	0,778	0,776	0,781	0,759	0,722	0,721	0,723
		K = 4	0,779	0,778	0,776	0,781	0,759	0,722	0,721	0,723
		K = 5	0,772	0,777	0,755	0,800	0,767	0,727	0,737	0,716
		K = 6	0,777	0,780	0,765	0,795	0,770	0,734	0,734	0,734
		K = 7	**0,794**	0,792	0,795	0,788	0,779	**0,747**	0,740	**0,754**
		K = 8	0,778	0,785	0,758	**0,814**	**0,781**	0,738	0,766	0,711
		K = 9	0,786	0,787	0,778	0,795	0,777	0,733	0,759	0,708
		K = 10	0,782	0,775	**0,797**	0,754	**0,781**	0,736	**0,768**	0,706

Table 5.1: Accuracy, F_1, Precision and Recall scores on RUMEN dataset for binary classification. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$.

In Table 5.2, the attention mechanisms presented in Chapter 4 are included to improve the input representation, in particular the patches representation. Indeed, as it seems that the patch alone is noisy as it does not focus on some specific words inside the patch, we believe that the attention mechanisms can lead to improved results. The $attentionMech_1$ applied on patches is first evidenced. So, our input

is the concatenation of weighted distributional representation of patches along with the $SimPatches(.,.)$ measure. Improved results are shown for both synonymy and hypernymy and for all k values. In fact, the maximum performance is achieved with $k = 9$ for the synonymy and with $k = 2$ for hypernymy. Comparing to the model with the same inputs but without attention, we can notice 4.7% improvement in Accuracy ($k = 9$) for synonymy model and 5.9% ($k = 2$) for hypernymy. Then, we introduce the second attention mechanism $attentionMech_2$, where we have two possible architectures of models : $seqArchitecture$ and $paralArchitecture$. For the $paralArchitecture$ architecture, no clear global improvement has occurred. This is probably due to the increase of the dimension of our input representation and the duplicated information we include. As for the $seqArchitecture$, we can observe improved results for almost all k values. In fact, applying the second attention mechanism on already weighted vectors boosts the classification process as it provides extra attention when looking jointly at the word pairs. The Accuracy and the F_1 score increase respectively with 2.5% and 2.4% ($k = 9$) for the first task and 1.6% ($k = 7$) and 0.6% ($k = 4$) for the second task over the model with only one attention. Note that this could also suggest an agglomerative cognitive process rather than a flat one. Indeed, some study in Cognitive Science evidence a tree-like structure of the brain for the acquisition of language.

	Inputs		Synonym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
RUMEN	Patches + SP	K = 1	0,796	0,798	0,789	0,807	0,765	0,731	0,723	0,738
	+ att_1	K = 2	0,816	0,817	0,809	0,824	**0,818**	**0,789**	0,792	**0,786**
		K = 3	0,813	0,810	0,818	0,803	0,807	0,774	0,781	0,768
		K = 4	0,817	0,817	0,810	0,825	0,811	0,780	0,785	0,775
		K = 5	0,798	0,799	0,789	0,809	0,806	0,772	0,784	0,760
		K = 6	0,806	0,805	0,803	0,808	0,810	0,778	0,786	0,769
		K = 7	0,819	0,816	0,827	0,805	0,812	0,776	0,798	0,756
		K = 8	0,827	0,824	0,834	0,814	0,802	0,758	0,802	0,719
		K = 9	**0,833**	**0,831**	**0,835**	**0,828**	0,809	0,769	**0,804**	0,737
		K = 10	0,827	0,825	0,829	0,820	0,812	0,774	0,803	0,748
	Patches + SP	K = 1	0,796	0,798	0,785	0,811	0,777	0,743	0,738	0,749
	+att_1 + att_2	K = 2	0,802	0,804	0,793	0,815	**0,811**	**0,780**	0,782	**0,779**
	paralArchi.	K = 3	0,807	0,808	0,801	0,815	0,800	0,768	0,771	0,764
		K = 4	0,806	0,804	0,809	0,799	0,808	0,777	0,780	0,775
		K = 5	0,804	0,806	0,792	**0,821**	0,809	0,774	0,793	0,757
		K = 6	0,806	0,807	0,798	0,815	0,802	0,766	0,783	0,750
		K = 7	0,814	0,815	0,807	0,823	0,804	0,767	0,786	0,750
		K = 8	0,822	0,820	0,825	0,815	0,789	0,746	0,778	0,716
		K = 9	**0,830**	**0,827**	**0,839**	0,814	0,808	0,767	**0,805**	0,733
		K = 10	0,813	0,811	0,814	0,808	0,807	0,769	0,797	0,744
	Patches + SP	K = 1	0,840	0,834	0,856	0,813	0,806	0,774	0,778	0,770
	+ att_1 + att_2	K = 2	0,848	0,845	0,854	0,837	0,805	0,766	0,794	0,739
	seqArchi.	K = 3	0,848	0,846	0,852	**0,840**	0,812	0,776	0,799	0,754
		K = 4	0,838	0,837	0,841	0,833	0,819	**0,783**	0,813	**0,756**
		K = 5	0,845	0,842	0,849	0,836	0,813	0,777	0,802	0,753
		K = 6	0,848	0,843	0,863	0,824	0,815	0,774	0,817	0,735
		K = 7	0,835	0,831	0,848	0,814	**0,820**	0,780	**0,826**	0,739
		K = 8	0,849	0,843	**0,876**	0,812	0,815	0,776	0,812	0,744
		K = 9	**0,855**	**0,851**	0,873	0,829	0,818	0,777	**0,826**	0,733
		K = 10	0,847	0,842	0,864	0,821	0,813	0,770	0,822	0,725

Table 5.2: Accuracy, F_1, Precision and Recall scores on RUMEN dataset for binary classification with attention mechanisms. Lexical split is applied. Note that SP stands for $SimPatches$, att_1 (resp. att_2) stands for $attentionMech_1$ (resp. $attentionMech_2$), paralArchi. for paralArchitecture and seqArchi. for seqArchitecture.

5.2.2 ROOT9 and Weeds evaluation

Two lexico-semantic relations exist in the ROOT9 and Weeds datasets: co-hyponymy and hypernymy. For this purpose, we evaluate these datasets together. For these two datasets, we proceed with the same experiments as the RUMEN dataset for

binary classification. For ROOT9, the results presented in Table 5.3 and Table 5.4 are globally similar to those obtained for the RUMEN dataset. However, we notice improvements of 8.1% ($k = 10$) in terms of F_1 score when we introduce patches as input for the co-hyponymy identification model and 2.2% ($k = 2$) for the hypernymy classification. For the rest, the result changes are globally similar to those with the RUMEN dataset. The *seqArchitecture* remains the best architecture to include attention in our models as we have an improvement of 3.4% of Accuracy ($k = 3$) for the co-hyponymy model and 3.5% of Accuracy ($k = 2$) for the hypernymy model, compared to models without attention and with the same input representation.

	Inputs		Co-hyponym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	Concat		0,907	0,938	0,949	0,888	0,852	0,895	0,901	0,927
	Concat + cos		0,911	0,941	0,951	0,931	0,835	0,883	0,891	0,874
	Patches	K = 1	0,904	0,937	0,938	**0,936**	0,848	0,892	0,902	0,881
		K = 2	0,894	0,929	0,937	0,922	**0,876**	**0,913**	0,908	**0,918**
		K = 3	0,888	0,926	0,937	0,915	**0,876**	0,911	0,926	0,897
		K = 4	0,894	0,929	0,943	0,915	0,863	0,904	0,901	0,907
		K = 5	0,895	0,930	0,944	0,917	0,866	0,904	0,920	0,888
		K = 6	0,904	0,937	0,944	0,929	0,873	0,908	**0,931**	0,886
		K = 7	0,684	0,938	0,948	0,927	0,855	0,894	0,927	0,863
		K = 8	0,903	0,935	**0,951**	0,920	0,850	0,892	0,904	0,881
		K = 9	0,903	0,936	0,946	0,926	0,848	0,892	0,902	0,881
		K = 10	**0,911**	**0,941**	0,948	0,935	0,850	0,893	0,904	0,881
	Concat + SP	K = 1	0,915	0,944	0,953	0,935	0,860	0,901	0,900	0,935
		K = 2	0,915	0,944	0,953	0,935	0,885	0,919	0,919	0,935
		K = 3	0,918	0,946	0,953	0,938	0,883	0,917	0,920	0,938
		K = 4	0,918	0,946	0,953	0,938	0,885	0,919	0,917	0,938
		K = 5	0,918	0,946	0,953	0,938	0,880	0,915	0,916	0,938
		K = 6	0,915	0,944	0,953	0,935	0,883	0,917	0,922	0,935
		K = 7	0,919	0,947	0,953	0,940	0,875	0,911	0,915	0,940
		K = 8	0,917	0,945	0,953	0,936	0,885	0,918	0,923	0,936
		K = 9	0,922	0,948	**0,955**	0,942	**0,893**	**0,924**	**0,927**	0,942
		K = 10	**0,926**	**0,951**	**0,955**	**0,947**	0,888	0,921	0,923	**0,947**
	Patches + SP	K = 1	0,917	0,945	0,942	**0,949**	0,767	0,909	0,911	0,907
		K = 2	0,912	0,942	0,945	0,940	0,896	0,927	0,918	**0,937**
		K = 3	0,912	0,942	0,945	0,940	0,896	0,927	0,918	**0,937**
		K = 4	0,912	0,942	0,945	0,940	0,896	0,927	0,918	**0,937**
		K = 5	0,914	0,943	0,951	0,935	0,891	0,923	0,919	0,928
		K = 6	0,916	0,945	0,948	0,942	**0,903**	**0,931**	0,938	0,923
		K = 7	0,915	0,944	0,952	0,936	0,881	0,914	0,939	0,890
		K = 8	0,918	**0,946**	0,950	0,942	0,878	0,911	**0,940**	0,883
		K = 9	0,918	**0,946**	0,950	0,942	0,870	0,905	0,933	0,880
		K = 10	**0,920**	**0,946**	**0,957**	0,936	0,873	0,908	0,934	0,884

Table 5.3: Accuracy, F_1, Precision and Recall scores on Root9 dataset for binary classification. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$.

	Inputs		Co-hyponym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9	Patches + SP	K = 1	0,921	0,948	0,946	0,950	0,880	0,915	0,912	0,919
	+ att_1	K = 2	0,921	0,947	0,949	0,947	0,908	0,936	0,925	0,946
		K = 3	0,933	0,956	0,961	0,950	**0,919**	**0,943**	0,938	**0,949**
		K = 4	0,931	0,955	0,962	0,947	0,911	0,937	0,943	0,930
		K = 5	0,930	0,954	0,956	0,952	0,913	0,938	0,940	0,937
		K = 6	0,933	0,956	0,959	0,952	0,918	0,941	**0,948**	0,935
		K = 7	0,934	0,956	0,959	0,954	0,916	0,940	0,944	0,937
		K = 8	0,937	0,958	0,958	0,959	0,903	0,931	0,941	0,921
		K = 9	**0,945**	**0,964**	0,961	**0,966**	0,893	0,923	0,935	0,912
		K = 10	0,940	0,960	**0,963**	0,957	0,891	0,922	0,937	0,907
	Patches + SP	K = 1	0,914	0,944	0,939	0,949	0,875	0,912	0,912	0,912
	+ att_1 + att_2	K = 2	0,912	0,943	0,939	0,947	0,903	0,932	0,919	**0,946**
	paralArchi.	K = 3	0,922	0,950	0,944	0,954	0,906	**0,934**	0,927	0,942
		K = 4	0,933	0,956	0,953	0,960	0,906	0,933	0,941	0,925
		K = 5	0,930	0,954	0,950	0,960	0,906	0,933	0,939	0,928
		K = 6	0,938	0,960	0,954	0,965	**0,908**	**0,934**	**0,943**	0,925
		K = 7	0,933	0,956	0,953	0,960	0,898	0,928	0,932	0,923
		K = 8	0,937	0,958	0,956	0,961	0,900	0,929	0,934	0,923
		K = 9	**0,950**	**0,967**	**0,965**	**0,970**	0,893	0,924	0,927	0,921
		K = 10	0,937	0,958	0,958	0,960	0,896	0,926	0,934	0,919
	Patches + SP	K = 1	0,937	0,958	0,958	0,960	0,911	0,938	0,929	0,946
	+ att_1 + att_2	K = 2	0,938	0,959	0,956	**0,963**	**0,931**	**0,952**	0,941	**0,963**
	seqArchi.	K = 3	**0,946**	**0,964**	0,968	0,961	0,930	0,950	0,945	0,956
		K = 4	0,945	0,963	**0,971**	0,956	0,916	0,941	0,940	0,942
		K = 5	**0,946**	**0,964**		0,957	0,910	0,936	0,935	0,937
		K = 6	0,943	0,963	0,968	0,957	0,904	0,932	0,939	0,925
		K = 7	0,942	0,962	0,968	0,957	0,894	0,924	0,946	0,902
		K = 8	0,940	0,961	0,966	0,956	0,891	0,921	**0,950**	0,893
		K = 9	0,937	0,958	0,970	0,947	0,885	0,916	0,943	0,891
		K = 10	0,935	0,957	**0,971**	0,943	0,885	0,916	0,948	0,886

Table 5.4: Accuracy, F_1, Precision and Recall scores on ROOT9 dataset for binary classification with attention mechanisms. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$, att_1 (resp. att_2) stands for $attentionMech_1$ (resp. $attentionMech_2$), paralArchi. for paralArchitecture and seqArchi. for seqArchitecture.

We observe the same improvements for the Weeds dataset in the Table 5.5 and the Table 5.6, when compared to RUMEN. The *seqArchitecture* is the best model for both co-hyponymy and hypernymy classification. In fact, major improvements of 8.2% in Accuracy and 19.7% in F_1 score ($k = 4$) for the co-hyponymy and 35.4% in Accuracy ($k = 7$) and 15.1% in F_1 score ($k = 9$) for the hypernymy classification.

To conclude, the impact of our input representation of patches and the attention mechanisms seems bigger for the Weeds dataset than for the ROOT9 dataset, but is at the same level of the RUMEN dataset.

			Co-hyponym vs Random				Hypernym vs Random			
	Inputs		Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds	Concat		0,718	0,429	0,414	0,444	0,810	0,422	0,433	0,412
	Concat + cos		0,789	0,587	0,548	0,632	0,842	0,527	0,530	0,524
	Patches	K = 1	0,684	0,383	0,358	0,413	0,789	**0,376**	0,374	0,377
		K = 2	0,688	0,388	0,364	0,416	0,767	0,349	0,329	0,372
		K = 3	0,664	0,379	0,338	**0,432**	0,764	0,335	0,318	0,353
		K = 4	0,686	0,383	0,359	0,410	0,765	0,357	0,330	**0,387**
		K = 5	0,689	0,365	0,355	0,375	0,770	0,350	0,333	0,367
		K = 6	0,680	0,369	0,347	0,394	0,768	0,360	0,336	**0,387**
		K = 7	0,684	0,393	0,386	0,400	0,773	0,355	0,339	0,372
		K = 8	**0,711**	**0,406**	**0,397**	0,416	0,782	0,350	0,351	0,348
		K = 9	0,702	0,389	0,379	0,400	0,790	0,367	0,372	0,363
		K = 10	0,693	0,392	0,370	0,416	**0,798**	0,347	**0,380**	0,319
	Concat + SP	K = 1	0,805	0,619	0,578	0,667	0,876	0,601	0,657	0,554
		K = 2	0,810	0,630	0,588	0,679	0,875	0,622	0,631	**0,613**
		K = 3	0,815	0,637	0,599	0,682	0,871	0,612	0,617	0,608
		K = 4	0,823	0,654	0,612	0,701	0,894	0,637	0,748	0,554
		K = 5	0,824	0,656	0,613	**0,705**	0,898	0,639	0,786	0,540
		K = 6	0,826	0,657	0,617	0,701	0,900	0,647	0,789	0,550
		K = 7	0,820	0,650	0,604	0,701	0,900	0,651	0,790	0,554
		K = 8	0,838	0,666	0,655	0,676	0,900	**0,653**	0,786	0,559
		K = 9	**0,841**	**0,671**	**0,660**	0,682	**0,901**	**0,653**	**0,796**	0,554
		K = 10	0,834	0,660	0,643	0,679	0,900	0,650	0,785	0,554
	Patches + SP	K = 1	0,755	0,521	0,486	0,562	0,767	0,524	0,551	0,5
		K = 2	0,788	0,560	0,554	0,565	0,856	0,557	0,576	0,540
		K = 3	0,788	0,560	0,554	0,565	0,856	0,557	0,576	0,540
		K = 4	0,788	0,560	0,554	0,565	0,856	0,557	0,576	0,540
		K = 5	0,764	0,542	0,503	0,587	0,863	0,565	0,607	0,530
		K = 6	0,786	0,575	0,544	0,610	0,866	0,570	0,617	0,530
		K = 7	0,780	0,573	0,531	0,622	0,861	0,572	0,591	0,554
		K = 8	0,786	0,580	0,543	0,622	0,870	0,582	0,627	0,544
		K = 9	**0,794**	**0,592**	**0,560**	0,628	0,871	**0,602**	0,623	**0,583**
		K = 10	0,780	0,580	0,530	**0,641**	**0,877**	0,598	**0,665**	0,544

Table 5.5: Accuracy, F_1, Precision and Recall scores on Weeds dataset for binary classification. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$.

	Inputs		Co-hyponym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds	Patches + SP	K = 1	0,798	0,571	0,578	0,565	0,873	0,581	0,652	0,524
	+ att_1	K = 2	0,823	0,624	0,632	0,616	0,886	0,629	0,696	0,573
		K = 3	0,851	0,665	0,715	0,622	0,899	0,639	0,796	0,534
		K = 4	0,804	0,604	0,581	0,628	**0,904**	0,666	**0,805**	0,568
		K = 5	0,852	0,667	0,718	0,622	0,903	0,668	0,792	0,578
		K = 6	0,838	0,653	0,664	0,641	0,901	0,663	0,776	0,578
		K = 7	0,844	0,659	0,685	0,635	**0,904**	**0,672**	0,793	**0,583**
		K = 8	**0,859**	**0,680**	**0,789**	0,629	0,878	0,654	0,760	0,573
		K = 9	0,833	0,643	0,655	0,632	0,903	0,668	0,783	**0,583**
		K = 10	0,845	0,664	0,686	**0,644**	0,900	0,663	0,768	**0,583**
	Patches + SP	K = 1	0,777	0,548	0,529	0,568	0,856	0,535	0,588	0,490
	+ att_1 + att_2	K = 2	0,801	0,579	0,584	0,575	0,867	0,598	0,609	0,588
	paralArchi.	K = 3	0,839	0,651	0,672	0,632	0,890	0,610	0,750	0,515
		K = 4	0,840	0,651	0,679	0,625	0,894	0,626	0,766	0,529
		K = 5	0,837	0,658	0,665	0,632	0,895	0,634	0,769	0,539
		K = 6	0,804	0,607	0,579	**0,638**	0,896	0,636	0,775	0,539
		K = 7	0,809	0,611	0,592	0,632	0,896	0,638	0,771	0,544
		K = 8	0,841	0,655	0,679	0,632	0,902	0,655	**0,801**	0,554
		K = 9	0,842	0,658	0,679	**0,638**	**0,903**	**0,661**	0,799	0,564
		K = 10	**0,861**	**0,685**	**0,743**	0,635	0,884	0,647	0,661	**0,632**
	Patches + SP	K = 1	0,828	0,641	0,635	0,647	0,916	0,703	0,864	0,593
	+ att_1 + att_2	K = 2	0,847	0,676	0,683	0,670	0,913	0,702	0,822	0,613
	seqArchi.	K = 3	0,863	0,702	0,725	0,679	0,917	0,714	0,856	0,613
		K = 4	**0,870**	**0,713**	**0,751**	0,679	0,923	0,736	0,872	0,637
		K = 5	0,868	0,711	0,746	0,679	0,922	0,732	0,861	0,637
		K = 6	0,860	0,698	0,718	0,679	0,922	0,734	0,856	0,642
		K = 7	0,855	0,696	0,694	**0,698**	**0,926**	0,747	**0,875**	0,652
		K = 8	0,853	0,693	0,691	0,695	**0,926**	0,751	0,861	0,667
		K = 9	0,852	0,689	0,689	0,689	**0,926**	**0,753**	0,856	0,671
		K = 10	0,863	0,698	0,736	0,663	0,923	0,752	0,824	**0,691**

Table 5.6: Accuracy, F_1, Precision and Recall scores on Weeds dataset for binary classification with attention. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$, att_1 (resp. att_2) stands for $attentionMech_1$ (resp. $attentionMech_2$), paralArchi. for paralArchitecture and seqArchi. for seqArchitecture.

5.2.3 Bless evaluation

For the Bless dataset, the lexico-semantic relations that need to be learned are meronomy and hypernymy (both asymmetric relations). Results of experiments are reported in Table 5.7 and the Table 5.8. Unlike the other datasets, we can notice

a considerable increase of F_1 score (17.1%) for the hypernymy learning model. An increase of 4.1% of Accuracy is also reported in the Table 5.7 comparing to the second experiment. Moreover, the introduction of patches gives similar results than without neighbors for the meronymy learning model. Then, we can clearly observe the impact of adding the $SimPatches(.,.)$ feature in our input as it boosts the classification process. By observing the results in Table 5.8, we lead to the same conclusion, that including the attention sequentially is better than in a parallel way. Compared to the models without attention, we have an improvement of 1.1% in terms of F_1 score for the meronymy model ($k = 6$) and an improvement of 5.9% in terms of F_1 score for the hypernymy model ($k = 2$).

	Inputs		Meronym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Bless	Concat		0,844	0,757	0,804	0,716	0,868	0,468	0,515	0,429
	Concat + cos		0,862	0,785	0,833	0,743	0,881	0,521	0,568	0,481
	Patches	K = 1	0,842	0,752	0,806	0,706	0,916	0,672	0,712	0,635
		K = 2	0,840	0,752	0,796	0,712	**0,922**	**0,692**	0,742	0,647
		K = 3	**0,848**	**0,764**	0,808	**0,725**	0,913	0,624	**0,743**	0,538
		K = 4	0,842	0,752	0,806	0,706	0,895	0,533	0,670	0,442
		K = 5	0,844	0,756	**0,813**	0,706	0,899	0,552	0,686	0,462
		K = 6	0,838	0,746	0,805	0,694	0,894	0,498	0,685	0,391
		K = 7	0,684	0,739	0,805	0,685	0,889	0,498	0,634	0,410
		K = 8	0,834	0,737	0,799	0,685	0,891	0,475	0,678	0,365
		K = 9	0,831	0,726	0,809	0,657	0,887	0,452	0,651	0,346
		K = 10	0,831	0,730	0,801	0,671	0,892	0,494	0,670	0,391
	Concat + SP	K = 1	0,877	0,811	0,853	0,774	0,904	0,602	0,683	0,538
		K = 2	0,877	0,811	0,849	0,776	0,905	0,623	0,669	0,583
		K = 3	0,879	0,816	0,849	0,785	0,913	0,653	0,712	0,602
		K = 4	0,878	0,814	0,850	0,781	**0,933**	**0,741**	0,780	**0,705**
		K = 5	0,882	0,820	0,853	0,789	0,929	0,717	0,776	0,667
		K = 6	0,884	0,824	0,851	0,799	0,932	0,735	0,783	0,692
		K = 7	**0,887**	0,828	0,860	0,799	**0,933**	0,739	**0,784**	0,698
		K = 8	**0,887**	**0,829**	0,857	**0,803**	0,912	0,653	0,695	0,615
		K = 9	0,883	0,823	0,849	0,799	0,915	0,671	0,704	0,641
		K = 10	0,881	0,818	0,853	0,785	0,917	0,676	0,714	0,641
	Patches + SP	K = 1	0,863	0,785	0,841	0,737	0,767	0,700	0,736	0,667
		K = 2	0,859	0,785	0,816	0,756	**0,939**	**0,759**	0,806	**0,718**
		K = 3	0,859	0,785	0,816	0,756	**0,939**	**0,759**	0,806	**0,718**
		K = 4	0,859	0,785	0,816	0,756	**0,939**	**0,759**	0,806	**0,718**
		K = 5	0,880	0,818	0,847	**0,791**	0,922	0,681	0,762	0,615
		K = 6	0,879	0,815	0,852	0,781	0,924	0,667	**0,815**	0,564
		K = 7	**0,881**	**0,818**	0,855	0,785	0,916	0,645	0,752	0,564
		K = 8	0,872	0,805	0,837	0,776	0,919	0,649	0,789	0,551
		K = 9	0,873	0,802	**0,857**	0,754	0,919	0,652	0,784	0,558
		K = 10	0,865	0,791	0,841	0,747	0,917	0,647	0,758	0,564

Table 5.7: Accuracy, F_1, Precision and Recall scores on Bless dataset for binary classification. Lexical split is applied. Note that SP stands for $SimPatches(.,.)$.

	Inputs		Meronym vs Random				Hypernym vs Random			
			Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Bless	Patches + SP	K = 1	0,863	0,788	0,831	0,750	0,926	0,713	0,743	0,686
	+ att_1	K = 2	0,867	0,795	0,834	0,760	0,939	0,763	0,807	0,724
		K = 3	0,892	0,834	0,878	**0,795**	**0,949**	**0,794**	**0,870**	**0,731**
		K = 4	0,891	0,831	0,884	0,783	0,928	0,698	0,807	0,615
		K = 5	0,892	0,831	0,889	0,779	0,932	0,715	0,830	0,628
		K = 6	0,890	0,826	0,894	0,768	0,932	0,717	0,825	0,635
		K = 7	0,890	0,826	0,896	0,766	0,930	0,701	0,826	0,609
		K = 8	0,890	0,826	0,892	0,770	0,932	0,706	0,841	0,609
		K = 9	0,896	**0,835**	**0,905**	0,776	0,934	0,714	0,864	0,609
		K = 10	**0,894**	0,832	0,904	0,769	0,932	0,707	0,854	0,602
	Patches + SP	K = 1	0,867	0,793	0,846	0,747	0,926	0,695	0,788	0,622
	+ att_1 + att_2	K = 2	0,880	0,814	0,863	0,770	**0,948**	**0,798**	0,838	**0,763**
	paralArchi.	K = 3	0,889	**0,835**	0,850	**0,820**	0,944	0,778	0,832	0,731
		K = 4	0,887	0,831	0,843	**0,820**	0,929	0,709	0,794	0,641
		K = 5	0,885	0,828	0,848	0,808	0,932	0,721	0,814	0,647
		K = 6	0,884	0,819	0,875	0,770	0,934	0,725	0,833	0,641
		K = 7	0,884	0,817	0,881	0,762	0,932	0,713	0,824	0,628
		K = 8	0,888	0,826	0,876	0,781	0,930	0,701	0,826	0,609
		K = 9	**0,893**	0,832	**0,891**	0,779	0,932	0,702	**0,853**	0,596
		K = 10	0,892	0,832	0,884	0,785	0,930	0,699	0,832	0,602
	Patches + SP	K = 1	0,894	0,832	0,910	0,766	0,945	0,776	0,854	**0,711**
	+ att_1 + att_2	K = 2	0,889	0,826	0,881	**0,777**	0,955	**0,822**	0,882	0,769
	seqArchi.	K = 3	0,886	0,817	0,902	0,747	**0,956**	0,821	0,907	0,75
		K = 4	0,888	0,821	0,899	0,756	0,946	0,769	**0,920**	0,660
		K = 5	0,891	0,825	0,911	0,754	0,942	0,751	0,894	0,647
		K = 6	**0,898**	**0,837**	0,915	0,772	0,941	0,746	0,893	0,641
		K = 7	0,889	0,820	0,916	0,743	0,942	0,747	0,908	0,635
		K = 8	0,882	0,811	0,891	0,745	0,940	0,733	0,922	0,609
		K = 9	0,887	0,815	**0,922**	0,731	0,934	0,703	0,9	0,577
		K = 10	0,889	0,818	0,920	0,737	0,932	0,683	0,914	0,545

Table 5.8: Accuracy, F_1, Precision and Recall scores on Bless dataset for binary classification with attention. Note that SP stands for $SimPatches(.,.)$, att_1 (resp. att_2) stands for $attentionMech_1$ (resp. $attentionMech_2$), paralArchi. for paralArchitecture and seqArchi. for seqArchitecture.

5.2.4 Balanced datasets

The RUMEN dataset is already balanced. For the three other datasets, we report in Table 5.9, the results of each input representation for the k value that contributed to the best result per model in previous tables. The reported results show simi-

lar tendencies to previous ones with unbalanced datasets. Indeed, the model with sequential attention remains the top-performing model when compared against baselines.

		Co-hyponym vs Random (k=3)				Hypernym vs Random(k=2)			
	Inputs	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
ROOT9_bal	Concat	0,872	0,876	0,889	0,863	0,839	0,829	0,809	0,85
	Concat + cos	0,877	0,881	0,890	0,872	0,847	0,834	0,836	0,831
	Patches	0,867	0,871	0,884	0,858	0,847	0,837	0,824	0,850
	Concat + SP	0,905	0,909	0,911	0,907	0,873	0,864	0,854	0,875
	Patches + SP	0,885	0,889	0,891	0,887	0,864	0,856	0,838	0,875
	Patches + att_1 + SP	0,897	0,902	0,902	0,902	0,879	0,870	0,860	0,881
	Patches + att_1 + att_2 + SP **paralArchi**.	0,897	0,901	0,910	0,892	0,870	0,861	0,853	0,868
	Patches + att_1 + att_2 + SP **seqArchi**.	0,921	0,922	0,943	0,902	0,905	0,896	0,899	0,894
		Co-hyponym vs Random (k=4)				Hypernym vs Random (k=7)			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Weeds_bal	Concat	0,571	0,581	0,561	0,602	0,653	0,645	0,631	0,659
	Concat + cos	0,597	0,604	0,586	0,623	0,672	0,665	0,649	0,681
	Patches	0,617	0,618	0,609	0,628	0,672	0,670	0,644	0,698
	Concat + SP	0,728	0,726	0,724	0,728	0,821	0,804	0,846	0,765
	Patches + SP	0,607	0,616	0,595	0,639	0,771	0,767	0,743	0,793
	Patches + att_1 + SP	0,757	0,753	0,756	0,748	0,848	0,832	0,881	0,788
	Patches + att_1 + att_2 + SP **paralArch**.	0,718	0,720	0,707	0,733	0,808	0,798	0,802	0,793
	Patches + att_1 + att_2 + SP **seqArchi**.	0,757	0,742	0,780	0,707	0,853	0,839	0,883	0,799
		Meronym vs Random (k=6)				Hypernym vs Random (k=2)			
	Algorithm	Acc.	F_1	Prec.	Rec.	Acc.	F_1	Prec.	Rec.
Bless_bal	Concat	0,814	0,836	0,853	0,819	0,906	0,929	0,929	0,929
	Concat + cos	0,833	0,854	0,866	0,842	0,911	0,933	0,929	0,937
	Patches	0,810	0,830	0,860	0,802	0,916	0,937	0,937	0,937
	Concat + SP	0,882	0,899	0,893	0,904	0,916	0,937	0,930	0,945
	Patches + SP	0,846	0,868	0,861	0,876	0,921	0,940	0,944	0,936
	Patches + att_1 + SP	0,856	0,875	0,880	0,870	0,918	0,938	0,944	0,933
	Patches + att_1 + att_2 + SP **paralArchi**.	0,853	0,873	0,871	0,876	0,929	0,947	0,941	0,952
	Patches + att_1 + att_2 + SP **seqArchi**.	0,872	0,884	0,931	0,842	0,908	0,929	0,954	0,905

Table 5.9: Accuracy, F_1, Precision and Recall scores on Bless dataset for binary classification. Note that SP stands for $SimPatches(.,.)$, att_1 (resp. att_2) stands for $attentionMech_1$ (resp. $attentionMech_2$), paralArchi. for paralArchitecture and seqArchi. for seqArchitecture.

5.3 Multi-task learning

For the multi-task learning, we report the results of the baseline work presented in Chapter 3 (without the pattern representations) in Table 5.10 for ROOT9 and Weeds datasets. These results prove that almost all multi-task architectures outperform the baseline binary classification ones and the best results on average are obtained for the Multiple Shared-private model. Therefore, we implement this model with our input representation (i.e. patches representation with attention mechanisms and $SimPatches(.,.)$ feature). The results are shown in Table 5.11 for ROOT9 and in the Table 5.12 Weeds datasets. For the ROOT9 dataset, we can notice that the performance of the model is enhanced for both tasks. However, for the Weeds dataset, the improvement is with a tiny margin on average. We believe that these models and architectures could lead to better results with some parameter tuning and optimization.

		Co-hyponym vs Random		Hypernym vs Random		Average Two Tasks	
	Algorithm	Acc.	F_1	Acc.	F_1	Acc.	F_1
ROOT9	Binary - Concat + cos	0,911	0,941	0,835	0,883	0,873	0,912
	Fully-shared	0,902	0,935	0,835	0,882	0,868	0,909
	Single Shared-private	0,916	0,945	0,860	**0,904**	0,888	0,924
	Multiple Shared-private	**0,922**	**0,949**	**0,865**	0,903	**0,893**	**0,926**
		Co-hyponym vs Random		Hypernym vs Random		Average Two Tasks	
	Algorithm	Acc.	F_1	Acc.	F_1	Acc.	F_1
Weeds	Binary - Concat + cos	0,789	0,587	0,842	0,527	0,815	0,557
	Fully-shared	0,810	0,627	**0,871**	0,616	0,841	0,621
	Single Shared-private	**0,825**	**0,667**	0,860	0,635	0,842	0,651
	Multiple Shared-private	0,824	0,657	0,865	**0,651**	**0,844**	**0,654**

Table 5.10: Accuracy, F_1, Precision and Recall scores on ROOT9 and Weeds datasets for two-task architectures.

	Algorithm		Co-hyponym vs Random		Hypernym vs Random		Average Two Tasks	
			Acc.	F_1	Acc.	F_1	Acc.	F_1
ROOT9	Multiple Shared-private	K = 1	**0,930**	**0,954**	0,875	0,911	0,902	0,933
		K = 2	0,929	0,953	0,904	0,934	**0,916**	**0,943**
		K = 3	0,916	0,945	0,899	0,929	0,908	0,937
		K = 4	0,918	0,946	**0,909**	**0,936**	0,914	0,941
		K = 5	0,922	0,948	0,893	0,925	0,907	0,937
		K = 6	0,916	0,945	0,893	0,923	0,905	0,934
		K = 7	0,923	0,949	0,888	0,919	0,906	0,934
		K = 8	0,927	0,952	0,889	0,921	0,908	0,937
		K = 9	0,922	0,948	0,885	0,917	0,903	0,933
		K = 10	0,926	0,951	0,889	0,921	0,908	0,936
	Multiple Shared-private **seqArchi.**	K = 1	**0,947**	**0,966**	0,913	0,939	0,930	0,952
		K = 2	0,938	0,959	0,916	0,941	0,927	0,950
		K = 3	0,942	0,962	**0,923**	**0,945**	**0,932**	**0,954**
		K = 4	0,930	0,954	0,909	0,935	0,934	0,944
		K = 5	0,934	0,956	0,903	0,930	0,918	0,943
		K = 6	0,934	0,956	0,908	0,934	0,921	0,945
		K = 7	0,941	0,961	0,899	0,927	0,920	0,944
		K = 8	0,937	0,958	0,896	0,924	0,916	0,941
		K = 9	0,937	0,958	0,913	0,937	0,925	0,947
		K = 10	0,941	0,961	0,894	0,923	0,917	0,942

Table 5.11: Accuracy and F_1 scores on ROOT9 dataset for Multiple Shared-private model with patches (seqArchi.).

	Algorithm		Co-hyponym vs Random		Hypernym vs Random		Average Two Tasks	
			Acc.	F_1	Acc.	F_1	Acc.	F_1
Weeds	Multiple Shared-private	K = 1	0,817	0,643	0,894	0,650	0,856	0,647
		K = 2	**0,848**	**0,686**	0,888	0,658	0,868	0,672
		K = 3	0,847	0,670	0,894	0,691	**0,871**	0,680
		K = 4	0,837	0,672	0,888	0,673	0,862	0,672
		K = 5	0,832	0,656	0,889	0,678	0,861	0,667
		K = 6	0,833	0,672	0,890	0,685	0,862	0,679
		K = 7	0,837	0,677	0,885	0,668	0,861	0,673
		K = 8	0,841	0,678	**0,898**	**0,696**	0,869	**0,687**
		K = 9	0,825	0,661	0,884	0,667	0,855	0,664
		K = 10	0,836	0,668	0,885	0,671	0,861	0,670
	Multiple Shared-private **seqArchi.**	K = 1	0,835	**0,678**	**0,909**	**0,722**	**0,872**	**0,700**
		K = 2	0,831	0,666	0,893	0,687	0,862	0,676
		K = 3	0,833	0,660	0,888	0,682	0,861	0,671
		K = 4	0,820	0,659	0,887	0,685	0,853	0,672
		K = 5	0,833	0,671	0,904	0,715	0,868	0,693
		K = 6	0,828	0,660	0,904	0,711	0,866	0,685
		K = 7	0,834	0,673	0,906	0,719	0,870	0,696
		K = 8	0,832	0,663	0,901	0,704	0,866	0,684
		K = 9	0,829	0,661	0,906	0,713	0,867	0,687
		K = 10	**0,836**	0,673	0,900	0,687	0,868	0,680

Table 5.12: Accuracy and F_1 scores on Weeds dataset for Multiple Shared-private model with patches (seqArchi.).

5.4 Pattern features

In order to extract the patterns between word pairs, we followed the same process as the baseline work introduced in 3. In fact, we downloaded the English Wikipedia dump and we extracted a large representative corpus of 14Gb. After pre-processing this corpus, we keep only the patterns that do not exceed a maximum length of 10 words. Since we include the k most similar words to our word pair in the input representation, we also include their patterns. For instance, if we have $k = 5$ neighbors for each word in the pair, 25 pairs are constructed and patterns are searched for each pair. Unfortunately, this process took more time than expected (more than one month) so, we didn't get the time to incorporate these patterns into our input representation. However, we introduce in Chapter 6, the main ideas we have to include these patterns and to encode them as well.

Chapter 6

Conclusions and Future Work

In our work, we presented a new way of defining the distributional and the pattern-based representations of learning models for the identification of lexico-semantic relations that hold between words. These representations are evaluated with both binary classification architectures and multi-tasking ones. Indeed, we follow the same architectures as a baseline work [Balikas 2019] but with new input configurations. The main idea behind our work is to explore the patch notion, used in the Computer Vision, in NLP models and applications.

The patch representation consists of including the distributional representations of the k most similar words (i.e. neighbors) in terms of cosine similarity measure into the input representation of word pairs. This way, $2 \times (k+1)$ embedding vectors (GloVe vectors of dimension 300 in our case) form the input of our classification models. We introduce as well a new similarity measure between patches, which is an extension to the cosine similarity measure between words.

Evaluation results over four gold-standard datasets (RUMEN, ROOT9, Weeds and Bless) prove the importance of this feature. As for the patches, attention mechanisms were necessary to guide the neural networks in distinguishing the valuable information from the extra large input vector fed. The *seqArchitecture* model outperforms the other models for both binary and multi-task classification.

We believe that including the adapted pattern-based representation to our work may benefit the classification performance. Indeed, once the patterns of the different pairs of words (all the possible combinations for words and their neighbors) are extracted, we can have multiple possibilities to encode them. Indeed, we can keep only the most frequent pattern for each pair of words or keep only the most frequent pattern for all pairs and get a single representation after using a BiLSTM encoding. In the first case, multiple representations for patterns are collected (a BiLSTM encoding for each pattern). We can get the average of these representations to include it to the input vector or the concatenation of all these representations directly.

Another possibility to do with respect of input representation is to encode individual words in patterns with contextual embeddings such as BERT [Devlin 2018] or ELMo [Peters 2018]. This may positively impact performance.

Since we couldn't conduct any conclusion regarding the right k value to choose, we believe that we need to include it as a parameter of the neural network to learn automatically, in particular for multi-task strategies. Optimization and tuning strategies need to be established in order to get the best results possible.

As for the attention mechanisms, other strategies than the PageRank algorithm could be used :

- Include the attention mechanism in the learning process of the neural networks architectures, to learn each vector importance automatically.
- Include both variable (learned) attention values and fixed ones (PageRank), similarly to a supervised process.

These attention mechanisms could also be used in LSTM or CNN architectures rather than using just the concatenation. Therefore, different configurations of LSTM and CNN architectures are to consider.

Finally, we would like to enhance the performance of our binary and multi-task classification strategies by integrating visual features. This will lead to a multimodal system. The underlying idea is to have a visual input representation for the word pair and its neighbors similarly to the textual representation. We can integrate an image for each word (e.g. from ImageNet [Russakovsky 2015]) as a visual feature. We can also look for a $SimPatches(.,.)$ notion in images (i.e. an image representing the pair of words or the background of an image containing these two words, etc.)

Our work and these future studies will soon be described in a publication intended for the European Conference on Information Retrieval (ECIR 2020)[1].

[1] https://ecir2020.org/

Bibliography

[Abadi 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org. 24

[Alkinani 2017] Monagi Alkinani and Mahmoud El-Sakka. *Patch-based models and algorithms for image denoising: a comparative review between patch-based images denoising methods for additive noise reduction.* EURASIP Journal on Image and Video Processing, vol. 2017, page 58, 08 2017. 2, 8

[Almeida 2019] Felipe Almeida and Geraldo Xexéo. *Word Embeddings: A Survey.* CoRR, vol. abs/1901.09069, 2019. 3

[Attia 2016] Mohammed Attia, Suraj Maharjan, Younes Samih, Laura Kallmeyer and Thamar Solorio. *CogALex-V Shared Task: GHHH - Detecting Semantic Relations via Word Embeddings.* In CogALex@COLING, 2016. 1, 7, 8

[Bahdanau 2014] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* CoRR, vol. abs/1409.0473, 2014. 2, 9

[Balikas 2019] Georgios Balikas, Gaël Dias, Rumen Moraliyski, Houssam Akhmouch and Massih-Reza Amini. Learning lexical-semantic relations using intuitive cognitive links, pages 3–18. 04 2019. 1, 7, 8, 10, 12, 13, 15, 16, 25, 44

[Baroni 2012] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do and Chung-chieh Shan. *Entailment Above the Word Level in Distributional Semantics.* In Proceedings of the 13th Conference of the European Chapter of the Association

for Computational Linguistics, EACL '12, pages 23–32, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. 7, 14, 15

[Bengio 2003a] Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Janvin. *A Neural Probabilistic Language Model*. J. Mach. Learn. Res., vol. 3, pages 1137–1155, March 2003. 4

[Bengio 2003b] Yoshua Bengio and Jean-Sébastien Senécal. *Quick Training of Probabilistic Neural Nets by Importance Sampling*, 2003. 4

[Benotto 2015] G. Benotto and Università di Pisa. Dipartimento di filologia letteratura e linguistica. Distributional models for semantic relations: A study on hyponymy and antonymy : Tesi di dottorato. 2015. 15

[Brin 1998] Sergei Brin and Lawrence Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In Seventh International World-Wide Web Conference (WWW 1998), 1998. 9, 21

[Chollet 2015] François Chollet*et al. Keras*. `https://keras.io`, 2015. 24

[Collobert 2008] Ronan Collobert and Jason Weston. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. 4

[Deerwester 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas and Richard A. Harshman. *Indexing by Latent Semantic Analysis*. JASIS, vol. 41, pages 391–407, 1990. 4

[Devlin 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. CoRR, vol. abs/1810.04805, 2018. 5, 45

[Dong 2017] Li Dong, Jonathan Mallinson, Siva Reddy and Mirella Lapata. *Learning to Paraphrase for Question Answering*. 08 2017. 1

[Eriguchi 2016] Akiko Eriguchi, Kazuma Hashimoto and Yoshimasa Tsuruoka. *Tree-to-Sequence Attentional Neural Machine Translation*. CoRR, vol. abs/1603.06075, 2016. 2, 9

[Fu 2015] Ruiji Fu, Jiang Guo, Yanyan Zhao, Wanxiang Che, Haifeng Wang and Ting Liu. *Learning Semantic Hierarchies: A Continuous Vector Space Approach*. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, pages 461–471, 2015. 7

[Gambhir 2016] Mahak Gambhir and Vishal Gupta. *Recent automatic text summarization techniques: a survey*. Artificial Intelligence Review, vol. 47, pages 1–66, 2016. 1

[Glavas 2017] Goran Glavas and Simone Paolo Ponzetto. *Dual Tensor Model for Detecting Asymmetric Lexico-Semantic Relations*. In EMNLP, 2017. 1, 7

[Guggilla 2016] Chinnappa Guggilla. *CogALex-V Shared Task: CGSRC - Classifying Semantic Relations using Convolutional Neural Networks*. In CogALex@COLING, 2016. 1, 7, 8

[Harris 1954] Zellig Harris. *Distributional structure*. Word, vol. 10, no. 23, pages 146–162, 1954. 3

[Hearst 1992] Marti A. Hearst. *Automatic Acquisition of Hyponyms from Large Text Corpora*. In COLING, 1992. 6

[Jiménez 2019] Sergio Jiménez, Fabio A. González, Alexander F. Gelbukh and George Dueñas. *word2set: WordNet-Based Word Representation Rivaling Neural Word Embedding for Lexical Similarity and Sentiment Analysis*. IEEE Computational Intelligence Magazine, vol. 14, pages 41–53, 2019. 9

[Joulin 2016] Armand Joulin, Edouard Grave, Piotr Bojanowski and Tomas Mikolov. *Bag of Tricks for Efficient Text Classification*. CoRR, vol. abs/1607.01759, 2016. 4

[Kathuria 2017] Neha Kathuria, Kanika Mittal and Anusha Chhabra. *A Comprehensive Survey on Query Expansion Techniques, their Issues and Challenges*. 2017. 1

[Kim 2014] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. CoRR, vol. abs/1408.5882, 2014. 8

[Kim 2017] Yoon Kim, Carl Denton, Luong Hoang and Alexander M. Rush. *Structured Attention Networks*. CoRR, vol. abs/1702.00887, 2017. 2, 9

[Kingma 2014] Diederik Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* International Conference on Learning Representations, 12 2014. 24

[Kozareva 2010] Zornitsa Kozareva and Eduard Hovy. *A Semi-supervised Method to Learn and Construct Taxonomies Using the Web.* In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP '10, pages 1110–1118, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. 6

[Lebret 2013] Rémi Lebret and Ronan Lebret. *Word Emdeddings through Hellinger PCA.* CoRR, vol. abs/1312.5542, 2013. 4

[Levy 2015] Omer Levy, Steffen Remus, Christian Biemann and Ido Dagan. *Do Supervised Distributional Methods Really Learn Lexical Inference Relations?* In HLT-NAACL, 2015. 15

[Lin 2017] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou and Yoshua Bengio. *A Structured Self-attentive Sentence Embedding.* CoRR, vol. abs/1703.03130, 2017. 2, 9

[Liu 2017] Pengfei Liu, Xipeng Qiu and Xuanjing Huang. *Adversarial Multi-task Learning for Text Classification.* CoRR, vol. abs/1704.05742, 2017. 1, 13

[Mikolov 2009] Tomas Mikolov, Jiri Kopecky, Lukas Burget, Ondrej Glembek and Jan ?Cernocky. *Neural Network Based Language Models for Highly Inflective Languages.* In Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '09, pages 4725–4728, Washington, DC, USA, 2009. IEEE Computer Society. 4

[Mikolov 2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan ernocký and Sanjeev Khudanpur. *Recurrent neural network based language model.* In INTERSPEECH, 2010. 4

[Mikolov 2013a] Tomas Mikolov, Kai Chen, Gregory S. Corrado and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space.* CoRR, vol. abs/1301.3781, 2013. 4

[Mikolov 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean. *Distributed Representations of Words and Phrases and Their*

Compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. 4

[Mnih 2007] Andriy Mnih and Geoffrey Hinton. *Three New Graphical Models for Statistical Language Modelling.* In Proceedings of the 24th International Conference on Machine Learning, ICML '07, pages 641–648, New York, NY, USA, 2007. ACM. 4

[Mnih 2008] Andriy Mnih and Geoffrey E. Hinton. *A Scalable Hierarchical Distributed Language Model.* pages 1081–1088, 01 2008. 4

[Mnih 2014] Volodymyr Mnih, Nicolas Heess, Alex Graves and Koray Kavukcuoglu. *Recurrent Models of Visual Attention.* CoRR, vol. abs/1406.6247, 2014. 2, 9

[Mrksic 2017] Nikola Mrksic, Ivan Vulic, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gasic, Anna Korhonen and Steve J. Young. *Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints.* CoRR, vol. abs/1706.00374, 2017. 7

[Nguyen 2017] Kim Anh Nguyen, Sabine Schulte im Walde and Ngoc Thang Vu. *Distinguishing Antonyms and Synonyms in a Pattern-based Neural Network.* CoRR, vol. abs/1701.02962, 2017. 1, 6, 7

[Pennington 2014] Jeffrey Pennington, Richard Socher and Christoper Manning. *Glove: Global Vectors for Word Representation.* volume 14, pages 1532–1543, 01 2014. 5, 24

[Peters 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer. *Deep contextualized word representations.* CoRR, vol. abs/1802.05365, 2018. 5, 45

[Ritter 2009] Alan Ritter, Stephen Soderland and Oren Etzioni. *What Is This, Anyway: Automatic Hypernym Discovery.* pages 88–93, 01 2009. 6

[Rocktäschel 2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský and Phil Blunsom. *Reasoning about Entailment with Neural Attention.* 09 2015. 2, 9

[Roller 2014] Stephen Roller, Katrin Erk and Gemma Boleda. *Inclusive yet Selective: Supervised Distributional Hypernymy Detection.* In COLING, 2014. 1, 7

[Roller 2018] Stephen Roller, Douwe Kiela and Maximilian Nickel. *Hearst Patterns Revisited: Automatic Hypernym Detection from Large Text Corpora.* CoRR, vol. abs/1806.03191, 2018. 6, 7

[Rush 2015] Alexander M. Rush, Sumit Chopra and Jason Weston. *A Neural Attention Model for Abstractive Sentence Summarization.* CoRR, vol. abs/1509.00685, 2015. 2, 9

[Russakovsky 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge.* International Journal of Computer Vision (IJCV), vol. 115, no. 3, pages 211–252, 2015. 45

[Santus 2015] Enrico Santus, Frances Yung, Alessandro Lenci and Chu-Ren Huang. *EVALution 1.0: an Evolving Semantic Dataset for Training and Evaluation of Distributional Semantic Models.* In LDL@IJCNLP, 2015. 15

[Santus 2016] Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu and Chu-Ren Huang. *Nine Features in a Random Forest to Learn Taxonomical Semantic Relations.* ArXiv, vol. abs/1603.08702, 2016. 1, 14, 15

[Shwartz 2016a] Vered Shwartz and Ido Dagan. *CogALex-V Shared Task: LexNET - Integrated Path-based and Distributional Method for the Identification of Semantic Relations.* CoRR, vol. abs/1610.08694, 2016. 1, 7, 8

[Shwartz 2016b] Vered Shwartz, Yoav Goldberg and Ido Dagan. *Improving Hypernymy Detection with an Integrated Path-based and Distributional Method.* CoRR, vol. abs/1603.06076, 2016. 1, 6, 7

[Snow 2005] R. Snow, D. Jurafsky and A. Y. Ng. *Learning syntactic patterns for automatic hypernym discovery.* In Proceedings of the Neural Information Processing Systems Conference (NIPS 2005), 2005. 1, 6, 7

[Snow 2006] Rion Snow, Daniel Jurafsky and Andrew Y. Ng. *Semantic Taxonomy Induction from Heterogenous Evidence.* In Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, ACL-44, pages 801–808, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. 6

[Turney 2010] Peter D. Turney and Patrick Pantel. *From Frequency to Meaning: Vector Space Models of Semantics.* CoRR, vol. abs/1003.1141, 2010. 4

[Vulic 2017] Ivan Vulic, Nikola Mrksic, Roi Reichart, Diarmuid Ó Séaghdha, Steve J. Young and Anna Korhonen. *Morph-fitting: Fine-Tuning Word Vector Spaces with Simple Language-Specific Rules.* CoRR, vol. abs/1706.00377, 2017. 7

[Vylomova 2015] Ekaterina Vylomova, Laura Rimell, Trevor Cohn and Timothy Baldwin. *Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning.* CoRR, vol. abs/1509.01692, 2015. 7

[Wang 2018] Qi Wang, Chenming Xu, Yangming Zhou, Tong Ruan, Daqi Gao and Ping He. *An Attention-based BI-GRU-CapsNet Model for Hypernymy Detection between Compound Entities.* 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pages 1031–1035, 2018. 1, 6

[Weeds 2004] Julie Weeds, David Weir and Diana Mccarthy. *Characterising Measures of Lexical Distributional Similarity.* 07 2004. 7, 14, 15

[Weeds 2014] Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir and Bill Keller. *Learning to Distinguish Hypernyms and Co-Hyponyms.* pages 2249–2259, 08 2014. 1, 7

[Wieting 2015] John Wieting, Mohit Bansal, Kevin Gimpel and Karen Livescu. *From Paraphrase Database to Compositional Paraphrase Model and Back.* CoRR, vol. abs/1506.03487, 2015. 7

[Zeng 2014] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou and Jun Zhao. *Relation Classification via Convolutional Deep Neural Network.* In COLING, 2014. 8