From lexical towards contextualized meaning representation

Diana-Nicoleta Popa

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Diana-Nicoleta POPA

Thèse dirigée par **Eric GAUSSIER**

préparée au sein du **Laboratoire d'Informatique de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Vers des représentations contextualisées de mots

From lexical towards contextualized meaning representation

Thèse soutenue publiquement le **27 septembre 2019**, devant le jury composé de :

Monsieur ERIC GAUSSIER
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Directeur de thèse
Monsieur ERIC VILLEMONTE DE LA CLERGERIE
CHARGE DE RECHERCHE, INRIA CENTRE DE PARIS, Rapporteur
Madame CLAIRE GARDENT
DIRECTRICE DE RECHERCHE, CNRS DELEGATION CENTRE-EST, Rapporteur
Monsieur LAURENT BESACIER
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Président
Monsieur ALEXANDRE ALLAUZEN
PROFESSEUR, UNIVERSITE PARIS-SUD, Examinateur
Monsieur JAMES HENDERSON
CHARGE DE RECHERCHE, INSTITUT DE RECHERCHE IDIAP - SUISSE, Examinateur
Monsieur JULIEN PEREZ
CHERCHEUR, NAVER LABS EUROPE - MEYLAN, Examinateur

Abstract

Continuous word representations (word type embeddings) are at the basis of most modern natural language processing systems, providing competitive results particularly when input to deep learning models. However, important questions are raised concerning the challenges they face in dealing with the complex natural language phenomena and regarding their ability to capture natural language variability.

To better handle complex language phenomena, much work investigated fine-tuning the generic word type embeddings or creating specialized embeddings that satisfy particular linguistic constraints. While this can help distinguish semantic similarity from other types of semantic relatedness, it may not suffice to model certain types of relations between texts such as the logical relations of entailment or contradiction.

The first part of the thesis investigates encoding the notion of entailment within a vector space by enforcing information inclusion, using an approximation to logical entailment of binary vectors. We further develop entailment operators and show how the proposed framework can be used to reinterpret an existing distributional semantic model. Evaluations are provided on hyponymy detection as an instance of lexical entailment.

Another challenge concerns the variability of natural language and the necessity to disambiguate the meaning of lexical units depending on the context they appear in. For this, generic word type embeddings fall short of being successful by themselves, with different architectures being typically employed on top to help the disambiguation. As type embeddings are constructed from and reflect co-occurrence statistics over large corpora, they provide one single representation for a given word, regardless of its potentially numerous meanings. Furthermore, even given monosemous words, type embeddings do not distinguish between the different usages of a word depending on its context.

In that sense, one could question if it is possible to directly leverage available linguistic information provided by the context of a word to adjust its representation. Would such information be of use to create an enriched representation of the word in its context? And if so, how can information of syntactic nature aid in the process? Further, what impact can this have on various natural language understanding tasks? One could thus investigate whether looking at the

representations of the words within a sentence and the way they combine with each-other can help build more accurate token representations for that sentence and thus facilitate performance gains on natural language understanding tasks.

In the second part of the thesis, we investigate one possible way to incorporate contextual knowledge into the word representations themselves, leveraging information from the sentence dependency parse along with local vicinity information. We propose syntax-aware token embeddings (SATokE) that capture specific linguistic information, encoding the structure of the sentence from a dependency point of view in their representations. This enables moving from generic type embeddings (context-invariant) to specific token embeddings (context-aware). While syntax was previously considered for building type representations, its benefits may have not been fully assessed beyond models that harvest such syntactical information from large corpora.

The obtained token representations are evaluated on natural language understanding tasks typically considered in the literature: sentiment classification, paraphrase detection, textual entailment recognition and discourse analysis. We empirically demonstrate the superiority of the token representations compared to popular distributional representations of words and to other token embeddings proposed in the literature.

The work proposed in the current thesis aims at contributing to research in the space of modelling complex phenomena such as entailment as well as tackling language variability through the proposal of contextualized token embeddings.

Acknowledgements

I would like to express my greatest gratitude to my PhD advisors for their invaluable support and guidance throughout this journey. To my PhD director Prof. Dr. Eric Gaussier and my industrial PhD advisors, Dr. James Henderson and Dr. Julien Perez for their availability, patience, clear explanations and inspiring vision. Working with you has been truly rewarding and I am extremely lucky and grateful for being given this chance.

I would like to thank Prof. Claire Gardent and Prof. Eric Villemonte de la Clergerie for taking the time to read and assess my PhD manuscript and for providing me with insightful scientific feedback. I would also like to thank Prof. Alexandre Allauzen and Prof. Laurent Besacier for accepting to be part of the PhD defense jury and evaluate my work and for the valuable discussions and suggestions.

I am thankful to my former advisors and my colleagues in Xerox Research Centre Europe, later become Naver Labs Europe and my colleagues from Laboratoire Informatique de Grenoble who have witnessed and encouraged my development throughout all this time. I am very grateful for being given support and feedback for my work as well as access to the computational resources that enabled this to exist. These have been wonderful and enriching working environments and I am forever grateful for the instructive conversations and exchanges I was part of.

Needless to say I would not be here today if it weren't for my dearest parents who have always been supportive, did their best for me and constantly encouraged me to aim higher. Your contribution to who I am today cannot be measured in words. Thank you! I am also thankful to my grand parents for their love and support.

Last, but not least, I am thankful to each and every one of my friends and dear ones who have provided me with their understanding, patience and endless support in my times of doubt and discouragement. Thank you for believing in me, I am very fortunate to have you near, no matter how far!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

1.1.1 Word representations

Continuous word representations (*word embeddings*) (Collobert and Weston, 2008; Turian et al., 2010; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013b; Pennington et al., 2014; Bojanowski et al., 2017) lie nowadays at the basis of most methods employed for natural language processing tasks. They encode richer representations than those provided by traditional symbolic or discrete models, removing the sparsity issues and enabling similarity judgments. Their usefulness has been proven in a variety of tasks such as paraphrase detection (Kenter and de Rijke, 2015), textual entailment recognition (Bowman et al., 2015), dependency parsing (Bansal et al., 2014), machine translation (Zou et al., 2013), language modelling (Peters et al., 2017) and so on. The created embeddings encode words as vectors such that words with similar meanings lie close to each other in the vector space, while dissimilar words are far apart. Vector similarity measures can then be used to approximate meaning similarity within this space.

At the core of most approaches to meaning representation and to embeddings creation lies the

distributional hypothesis (Harris, 1954). This represents the idea that words that appear in similar contexts tend to bear similar or related meanings, encouraging thus the implementation of models that look at a word's context to infer the word's meaning. This enables leveraging the available large amounts of data to create continuous word representations through unsupervised methods (Mikolov et al., 2013b; Pennington et al., 2014).

However, while generic word representations are employed for a wide range of tasks, further work has looked into adjusting these word embeddings to satisfy various linguistic constraints. This is considered to be necessary due to the need to distinguish between semantic similarity and other types of semantic relatedness (Hill et al., 2015), that is not possible when depending on purely distributional knowledge (Glavaš and Vulić, 2018). Consequently, the created word representations are either better tailored for solving a particular task, such as the lexical entailment-specific embeddings of Vulic and Mrksic (2018) or can be used to obtain improved results invariably of the task (Yu and Dredze, 2014).

A common tendency to enable the creation of enriched word representations, is to leverage the existence of multiple language resources, either during the embeddings creation phase or as a post-processing step. In that sense, Faruqui et al. (2015) propose leveraging information from semantic lexicons such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998) and the Paraphrase Database (Ganitkevitch et al., 2013) to refine word embeddings representations in a post-processing step denoted as *retrofitting*. Yu and Dredze (2014); Xu et al. (2014) propose including linguistic constraints through regularization terms included during the training of word embeddings. Mrkšic et al. (2016) propose a model that injects antonymy and synonymy constraints into vector space representations which are further evaluated on semantic similarity and dialogue state tracking. Mrkšić et al. (2017) leverage cross-lingual resources for improving the quality of word representations, while Glavaš and Vulić (2018) propose a method to learn a specialization function that can be further applied to previously unseen words.

Furthermore, when going beyond tasks that rely on meaning relatedness, into recognizing logical relations such as hypernymy, entailment or contradiction, using only distributional semantic

models has been proven to be challenging and even insufficient (Levy et al., 2015; Kruszewski et al., 2015). More particularly, for the case of lexical inference, Levy et al. (2015) show that many supervised models learn whether one of the words in a pair of words is a "prototypical hypernym" (to be read as "it tends to entail", rather than "it tends to be entailed"), instead of learning a concrete relation that holds between the two given words. This finding is consistent across several methods for word representations creation, taking into account different context types. Therefore, further effort has been put into better modelling such relations. For this, some work creates hypernymy-specific word embeddings (Yu et al., 2015; Nguyen et al., 2017; Chang et al., 2018), while other work goes further into modelling entailment within the vector space itself through information inclusion (Kruszewski et al., 2015; Henderson and Popa, 2016). Nevertheless, such methods contribute to the space of word representations, obtaining competitive results on the respective tasks.

1.1.2 Modelling compositionality

Although modelling the individual words is important, representing and reasoning about longer units of text such as phrases, sentences or paragraphs requires going beyond isolated word representations. This aspect has been typically handled in various ways: one possibility is to extend methods that initially dealt with word representations to consider longer units of text (Le and Mikolov, 2014; Kiros et al., 2015) as part of an unsupervised approach. Another option is to explicitly model in some way compositionality over individual word representations through various mathematical operations (Clark et al., 2008; Mitchell and Lapata, 2010). Other work aims to learn composition operators that map word representations to representations of longer units of text, often employing neural network architectures (Hochreiter and Schmidhuber, 1997; Socher et al., 2013b; Kalchbrenner et al., 2014; Kim, 2014). However, the resulting sentence representations are learned as part of a supervised task, making the learned representations tuned for the task. To account for that, some work has also looked at learning universal sentence representations (Conneau et al., 2017; Cer et al., 2018), as part of transfer learning setups.

Most often though, such methods still assign one single vector per sentence which is arguably insufficient for containing the entire meaning of the sentence.

As recent work (Dasgupta et al., 2018) shows though, the proposal in Conneau et al. (2017) does not exhibit significant compositionality and, in general, many models leveraging such sentence representations rely on heuristics of the data for achieving high performance. For the SNLI dataset (Bowman et al., 2015) for example, a high overlap in words between premise and hypothesis has been proven to be predictive of an entailment relation between sentences, while the presence of negations and little word overlap indicates potential contradiction.

Nevertheless, while multiple approaches exist, no consensus has been reached regarding the best method to employ for obtaining compositionality without losing information captured in the individual word representations, yet generalizing to encoding the entirety of a sentence and of the dependencies within it.

According to the Principle of Compositionality, also known as Frege's principle (Frege, 1884), the meaning (semantic intepretation) of a complex expression is a function of the meanings of its parts and of the rules by which they are combined (Bach, 1989; Partee et al., 1990), with Partee (1995) further suggesting that these rules could be of syntactic nature. That is to say, finding a way to combine the meaning of individual words into longer units of texts such as phrases or sentences is crucial to achieving overall text understanding. Such an approach implies that the interpretation of a sentence is the combination of lexical semantics and syntactical aspects within the sentence, supporting that the same words can be combined differently to provide different meanings, like in the below example from Landauer et al. (1997):

- *It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.*

- *That day the office manager, who was drinking, hit the problem sales worker with the bottle, but it was not serious.*

In this respect, much work has investigated methods to combine the vectorial representations of words into representations of the phrases and sentences they form. Some early work questioned the necessity of taking into account word order by representing a passage of text as a linear combination of its word vectors (Landauer et al., 1997; Foltz et al., 1998), while other work argued for the importance of considering syntactic information (West and Stanovich, 1986). In general the reduced complexity of using an averaging approach along with the benefit of maintaining the same dimensionality in the output as that of the input representations may be preferred for some applications (Landauer et al., 1997; Iyyer et al., 2015). Yet, whenever there is a need for encoding word order or distinguishing between representations of two passages of text sharing the same vocabulary, but differing in meaning (like in the previous example), leveraging the syntactic structure could constitute an alternative. In particular, it has been shown that models of semantic similarity should encourage the combination of semantic content in a syntactically-aware manner (Mitchell and Lapata, 2010).

Starting from a vector-based representation for each word, the meaning of sequences of words is often obtained through linear combinations of the input vectors, such as addition or weighted average. Variations also exist over such additive models, as that proposed by Kintsch (2001) to model predicate-argument structures, relying on the idea that one can infer the meaning of a predicate depending on the argument it operates upon. In such a model, the neighbours of target words and their arguments are considered, allowing for distinguishing cases such as *the cat ran away* and *the color ran*.

Formally, Mitchell and Lapata (2008, 2010) propose a framework for the use of additive and multiplicative functions (and their combinations) to perform vector composition. Their framework allows for integrating knowledge about syntactic relations as well as any additional world knowledge or information about the neighbours of target words. It also allows for weighing differently the individual contributions of the words to the final sequence representation. The proposed operators are applied to word vectors obtained through different methods: either based on co-occurrence statistics with neighbouring words or derived through topic modelling. Perfor-

mance is assessed on sentence similarity rating, with multiplicative models obtaining competitive results on the word co-occurrence-based semantic space and additive models performing well on the topic-based semantic space. Many subsequent models use their framework or variations of it as starting point for their compositional approaches (Erk and Padó, 2008; Baroni and Zamparelli, 2010; Zanzotto et al., 2010; Kartsaklis and Sadrzadeh, 2013), including when integrated in deep neural networks (Iyyer et al., 2015).

An alternative to using vector addition is represented by tensor products. Following Smolensky (1990) in integrating the symbolic and distributional approaches, Clark and Pulman (2007) propose using tensor products over the traversal of syntactic dependency trees. They consider both word representations and the syntactic structures they appear in: vectors representing the meaning of words are combined in a tensor product with vectors representing their roles. It is interesting to note, though, that their modelling of the vectors for syntactic relations is regarded as an open question and handled simply by assigning them pre-defined values that would satisfy some constraints. The created representations obtained from their proposed tensor products do allow for distinguishing sentences with the same vocabulary but different meanings from each other, such as:

- *The cat chased the dog.*
- *The dog chased the cat.*

enabling the difference between *cat* being an agent in the first sentence and a patient in the second. One of the main issues, however, with tensor products lies in the exponential growth of the space created by the combination of such representations. A solution for this aspect can be obtained by projecting the tensor product back into the space of the original vectors by applying circular convolutions (Plate, 1991). However, later work (Mitchell and Lapata, 2010) showed that the projection of the tensor products into a lower dimensional space provides a decreased performance and, furthermore, in general the tensor product based approaches are outperformed by the multiplicative ones. Clark et al. (2008) also make use of tensor products to

propose linking models of linguistic composition to vector composition, while Widdows (2008) provide an analysis of different compositional models based on vector operations.

Vectors are not the only structure used to encode word representations as part of modelling compositionality. Baroni and Zamparelli (2010) propose a model where nouns are encoded with vectors and adjectives with matrices that act as functions over the nouns. Their model is evaluated only on adjective-noun composition, not providing a framework of composition for a full sentence. Rudolph and Giesbrecht (2010) propose a Compositional Matrix-Space model, proving theoretically its appropriateness for modelling compositionality. Yessenalina and Cardie (2011) also propose modelling all words as matrices and their interactions as matrix multiplications. However, due to the associative property of matrix multiplication such an approach fails to model syntactic differences or to clearly delimit the scope of negations. Grefenstette and Sadrzadeh (2011) model relational words (adjectives, adverbs, verbs) as matrices and their arguments (nouns) as vectors within a categorical model that focuses on subject-verb-object triples.

Some work uses models from the family of recursive neural networks to achieve competitive results on several language understanding tasks by relying on the structure of natural language. Within these approaches, compositionality is considered over the binarized constituency parse tree of a sentence, by having the representation for any internal node in the tree computed as a nonlinear function of the representations of its children. Socher et al. (2011b) model phrases and sentences using recursive autoencoders, while Socher et al. (2012) propose using recursive neural networks for modelling compositionality, by considering both a vectorial and a matrix representation for each constituent in the parse tree. This enables the modelling of both the content of each word (through the vector) and of its interaction with neighbouring words (through the matrix). Socher et al. (2013a) extend this further into a more powerful recursive neural tensor network that includes a tensor product.

Tai et al. (2015) propose Tree-LSTMs, a generalization over the standard long short-term memory architecture (Hochreiter and Schmidhuber, 1997) to tree-structured topologies. They lever-

age both dependency and constituency parses and show such architectures bring improvements on the tasks of semantic relatedness and sentiment classification. Similarly, Wang et al. (2017) obtain improvements on classifying implicit discourse relations using Tree-LSTM and Tree-GRU architectures, while Irsoy and Cardie (2013) propose using bidirectional recursive neural networks over binary parse trees. Much other work leverages the structure of parse trees, sometimes combined with an attention mechanism (Yao Zhou and Pan, 2016; Kokkinos and Potamianos, 2017), implicitly obtaining some form of compositionality.

1.1.3 Compositionality and contextuality

Apart from relying on the way the words combine with each-other and in order to be able to reason beyond single lexical units by considering longer units of text such as sentences and phrases, one also needs to take into account the different possible meanings of words. And one possible (and often preferred) way to disambiguate the meaning of a word is by looking at its immediate neighbouring context.

While contextual information, be it linear (Mikolov et al., 2013b) or of syntactic nature (Levy and Goldberg, 2014), is much used in distributional semantics models, the obtained representation of a word encodes an abstraction over the occurrences and uses of that word in the dataset it is learned from (Westera and Boleda, 2019). A distinction is thus to be made between the general *word type embedding* which denotes the representation a word has across different contexts and a specific *word token embedding*, denoting the representation of a word in a particular context. While, as previously discussed, the generic word type embeddings constitute an important and often employed starting point within many natural language understanding tasks, it is the particular use of a given word that best disambiguates its meaning. Therefore, to get an even clearer understanding of the role the word plays and its contribution to the overall meaning of a particular sentence, one should ideally take the full sentence into account and focus on the in-context representation of that word.

Compositionality is thus inevitably linked to contextuality. As Frege (1884) points out in the Context principle, that later Wittgenstein (1953) adheres to, the meaning of a word should be considered only in the context of a statement and thus in its use. That is to say the meaning of the whole statement is constructed from its words (compositionality), yet the meaning of the words is derived from the entirety of the statement (contextuality). This suggests that a model of compositionality should be seen as one that promotes contextual awareness just as a model of contextuality should enable composition of meaning.

In that respect, the proposal in Chapter 3, also reflected in Popa et al. (2019a,b), aims at enabling the construction of word representations taking into account the context they appear in and the way the compose with each other. Context is considered from both a local linear perspective as well as a syntactic one. Locally, this is achieved by inducing information from the neighbours of a target word into the word representation itself. From a syntactic perspective, the parse tree of the sentence is encoded, by conditioning the representation of a target word with respect to representations of all the words it is connected to in the syntactic graph, considering the different types of connections as different relations between words. Since it is possible to reach any word from any other word within the syntactic graph of a sentence, information can "flow" from one word to another through the relations, resulting in each word representing a compressed view of the graph from its own perspective. In this model, contextuality becomes thus encoded within each word's representation (Popa et al., 2019a,b).

Much previous work relied on syntactic information for computing the word vector representations themselves (Padó and Lapata, 2007; Baroni and Zamparelli, 2010; Baroni and Lenci, 2010; Levy and Goldberg, 2014; Weir et al., 2016; Komninos and Manandhar, 2016; MacAvaney and Zeldes, 2018). But most of this work provided general purpose word type embeddings computed on the basis of syntactic contexts across large amounts of data, rather than *token embeddings* that encode the full syntactic graph of each individual sentence they are part of. Thus in such cases modelling compositionality and contextuality still need to be tackled within the parameters of an architecture built on top of these representations.

An approach that does include syntactic information into word representations and also goes beyond models that use tree-based recursive architectures to obtain compositionality, as those presented in Section 1.1.2, is that of Erk and Padó (2008). They propose building a structured vector space that provides for each word instance a vector denoting the lexical meaning of the word as well as a set of vectors denoting the selectional preferences the word has for its argument positions across multiple relations, thus inducing some form of syntactically aware contextual representations. Selectional preferences had been previously used for a variety of tasks from syntactic disambiguation (Hindle and Rooth, 1993) and word sense disambiguation (McCarthy and Carroll, 2003) to semantic role labeling (Gildea and Jurafsky, 2002), but mostly verbs were considered as taking part in relations. The obtained vector representations of Erk and Padó (2008) are further combined according to the framework of Mitchell and Lapata (2008). Nevertheless, Erk and Padó (2008) acknowledged the limitations of their proposed model to allow for information to "flow" from one word to another as relations are modeled in isolation and not within the graph of a sentence as in the current proposal. Additionally, in their proposed framework the meaning of a particular word is neither computed by integrating information from multiple relations as in the current proposal nor does it leverage distributional information computed from large amounts of data.

1.2 Outline of the thesis

As a first step, we focus on the representation of basic lexical units, namely the words, and propose a framework for including the entailment notion within the vector space itself, thus contributing to the space of word representations (Henderson and Popa, 2016). For that we use an instance of lexical entailment, namely hyponymy detection as a proxy task to modelling information inclusion. The relationship is modeled between lexical units disregarding the context they appear in, relying solely on the probability of features being "known" or "unknown" within the different dimensions of their representations. A mean-field approximation to entailment between binary vectors is thus proposed, along with operators that measure entailment between

vectors. Further, we show that the proposed framework can be seen as an approximation of a popular distributional semantics model and evaluate the potential of the proposal on the task of hyponymy detection.

The context of the proposal, its motivation as well as the necessary introductory notions are presented in Section 2.1. Section 2.2 presents the related work on hyponymy detection, with a focus on methods that are close to the current proposal through their aim to learn hypernymy-specific representations or to encode entailment. The proposal to model entailment in a vector space is described in Section 2.3, while Section 2.4 provides a way to reinterpret an existing model of distributional semantics as approximating one of semantic entailment. The experimental setup, the evaluation of this interpretation as well as that of the entailment operators are provided in Section 2.5. Some concluding remarks are presented in Section 2.6.

The next natural step is to abstract away from the lexical unit view and aim for a model of contextuality. As compositionality and contextuality are interlinked, encoding words within their context, while also taking into account the role each word plays in the sentence, could serve as a starting point for methods that aim at modelling compositionality. This leads to the proposal in Chapter 3. Both linear and syntactic contexts within a sentence are leveraged to compute the SATokE *token embeddings*, syntactically-informed word representations adjusted to their context (Popa et al., 2019a,b). Contextuality is achieved in this model as the context is encoded within the word vectors themselves and not within an architecture set up on top of them, which is often task dependent. The obtained token embeddings are computed such as to provide a view of the graph of the entire sentence from their perspective. The method used to induce these representations is based on tensor factorization which has proven to be successful at modelling latent representations of entities and relations in graphs (Nickel et al., 2012; Trouillon et al., 2016). The obtained token embeddings are tested on a wider range of tasks, going beyond just textual entailment: sentence sentiment classification, question type classification, paraphrase detection, implicit discourse relation classification.

Related work on context-aware token embeddings is presented in Section 3.1 in comparison to

the current proposal. The algorithm for token embeddings computation is detailed in Section 3.2, while the overall experimental protocol along with the implementation details are presented in Section 3.3. Details of the different tasks considered for evaluation as well as the obtained results are presented in Section 3.4. Finally, Section 3.5 presents the application of the proposed token embeddings to the task of implicit discourse relation classification, while Section 3.6 concludes the chapter.

Some final remarks, a summary of the achievements of the current thesis and possible future directions are outlined in Chapter 4.

Chapter 2

A vector space for the distributional semantics for entailment

2.1 Introduction

A widely studied fundamental issue in computational semantics has been that of modelling the entailment relation between two texts (Dagan et al., 2006; Mirkin et al., 2009; Rocktäschel et al., 2016; Chen et al., 2017). Its importance stems from its potential applicability in many natural language processing subfields where one needs to verify that some input text entails some output text, like for example in question answering (Sacaleanu et al., 2008), machine translation (Padó et al., 2009) or abstract summarization (Pasunuru et al., 2017). The notion of textual entailment is based on the idea that the same textual meaning can be expressed in different ways. It is generally defined as the directional relationship between two text fragments: the entailing *Text (T)* and the entailed *Hypothesis (H)* . An entailment relationship is said to hold between T and H if, by reading T, a human would infer that H is most likely true (Dagan et al., 2006).

While there has been a lot of interest and significant recent advances in modelling entailment within a vector-space (Bowman et al., 2015; Camburu et al., 2018; Kim et al., 2019), most of

$\Rightarrow$	*unk*	f	g	$\neg f$
unk	1	0	0	0
f	1	1	0	0
g	1	0	1	0
$\neg f$	1	0	0	1

Table 2.1: Pattern of logical entailment between nothing known (*unk*), two different features f and g known, and the complement of f ($\neg f$) known.

the existing work does not explicitly create a vector space of entailment, but rather encodes the relationship in the parameters of a classifier. As Levy et al. (2015) pointed out, for the simpler case of lexical entailment, such an approach has the risk of only learning to detect the generality of one of the terms involved in the relationship, rather than the actual relationship between the two terms. In contrast to this, the current work proposes a new framework for modelling entailment in a vector-space such that the entailment notion is captured within the vector space itself as outlined in Henderson and Popa (2016). To illustrate its effectiveness, a distributional-semantic model of hyponymy detection is used as an initial use case.

Identifying hyponymy relations is considered to be useful for a series of tasks ranging from taxonomy creation (Snow et al., 2006) to recognizing textual entailment (Dagan et al., 2013). Hyponymy/hypernymy represents the directional relation between a generic, superordinate term (denoted as the *hypernym*) and a specific, subordinate instance of it (denoted the *hyponym*). More precisely, it represents a *type-of* relationship between two terms or phrases, with the hyponym necessarily implying the hypernym, but not vice versa. For example, “dog”, “cat”, “parrot”, “rabbit” etc are all hyponyms of the word “animal”, which represents their hypernym. Hyponymy is thus the canonical type of lexical entailment, with words representing the basic lexical units between which such a relationship can hold. Reasoning about entailment implies thus being able to reason about hyponymy relations.

Differently from previous vector-space models of entailment, the current work provides a framework in which it is possible to explicitly model what information is *known* and *unknown*, offering a change of dichotomy from “true” - “false” to “known” - “unknown”. This represents a crucial property, because the entailment relationship fundamentally reflects information in-

clusion, that is, what information is known and what is not known: a representation y entails a representation x if and only if everything that is known given x is also known given y. Thus, entailment is modeled in a vector space by representing in each dimension something that is possibly known. This is illustrated in Table 2.1: knowing that a feature f is true always entails knowing that same feature, but never entails knowing that a different feature g is true. Also, knowing that a feature is true always entails not knowing anything (*unk*), since strictly less information is still entailment, but the reverse is never true. Table 2.1 also illustrates that knowing that a feature f is false ($\neg f$) patterns exactly the same way as knowing that an unrelated feature g is true.

Previous vector-space models have been very successful at modelling semantic similarity, in particular using distributional semantic models (e.g. (Deerwester et al., 1990; Schütze, 1993; Mikolov et al., 2013a)). Distributional semantics uses the distributions of words in contexts to induce vector-space embeddings of words, which have been shown to be useful for a wide variety of tasks. In such models two words are predicted to be similar if the dot product between their vectors is high. But the dot product is a symmetric operator, which makes it more natural to interpret these vectors as representing whether features are true or false, whereas the dichotomy "known" versus "unknown" is asymmetric. This could be a reason why modelling lexical entailment with distributional semantic models has been challenging (Levy et al., 2015).

To develop a vector-space model of whether features are known or unknown, we start with discrete binary vectors, where 1 means "known" and 0 means "unknown". As illustrated in Table 2.1, entailment between these discrete binary vectors can be calculated by independently checking each dimension and then taking the conjunction. However, as soon as we try to do calculations with *distributions* over these vectors, we need to deal with the case where the features are not independent. For example, if feature f has a 50% chance of being true and a 50% chance of being false, we can't assume that there is a 25% chance that both f and $\neg f$ are known. This simple case of mutual exclusion is just one example of a wide range of constraints between features which need to be handled in semantic models. These constraints mean that

the different dimensions of the vector space are not independent, and therefore exact models are not factorised. Because the models are not factorised, exact calculations of entailment and exact inference of vectors are intractable.

A popular approach to perform efficient inference for intractable models is through the use of mean-field approximations. In a mean-field approximation, distributions over binary vectors are represented using a single probability for each dimension. These vectors of real values represent the basis of the proposed vector space for entailment (Henderson and Popa, 2016).

In the following the related work is discussed in Section 2.2, while the vector-space model which provides a formal foundation for a distributional semantics of entailment is proposed in Section 2.3. The framework is derived from a mean-field approximation to entailment between binary vectors, and includes operators for measuring entailment between vectors, and procedures for inferring vectors in an entailment graph. The obtained word vectors represent the probabilities of features being known or unknown. The framework is further validated in Section 2.4 by using it to reinterpret existing distributional semantics word embedding vectors (Word2Vec (Mikolov et al., 2013a)) as approximating an entailment-based model of the distribution of words in contexts. This reinterpretation enables the usage of existing word embeddings as an unsupervised model of lexical entailment, successfully predicting hyponymy relations using the proposed entailment operators in both unsupervised and semi-supervised experiments. The experimental setup and the results of hyponymy detection are presented in Section 2.5, while Section 2.6 outlines the main contributions of the proposal.

2.2 Related work

Since the evaluation of lexical entailment is limited in the current work to that of hyponymy / hypernymy, not including other related lexical relations (cf. (Weeds et al., 2014; Vylomova et al., 2015; Turney and Mohammad, 2014; Levy et al., 2014)), the presentation of related work will also be restricted to that of hyponymy detection. More complex cases of entailment are left to

future work on compositional semantics.

There has been a significant amount of work on using distributional-semantic vectors for hyponymy detection, using supervised, semi-supervised or unsupervised methods (e.g. (Yu et al., 2015; Necsulescu et al., 2015; Vylomova et al., 2015; Weeds et al., 2014; Fu et al., 2015; Rei and Briscoe, 2014)). However, most of this work uses measures computed outside the vector space for entailment detection, applied mostly to traditional feature-based word vectors. Such measures include: symmetric measures like LIN (Lin, 1998), asymmetric measures such as WeedsPrec (Weeds and Weir, 2003; Weeds et al., 2004), balAPinc (Kotlerman et al., 2010), invCL (Lenci and Benotto, 2012) and entropy-based measures like SLQS (Santus et al., 2014). Later, Shwartz et al. (2017) provide a comprehensive review study of such measures across different distributional semantic spaces. Other work encodes hyponymy into the parameters of a classifier (Baroni et al., 2012; Roller et al., 2014; Fu et al., 2015). Since the current proposal differs from the above-mentioned, in that the modelling of entailment is performed within the vector space, not outside it and not in the parameters of a classifier, no thorough comparison is provided to these methods.

Among the methods that learn hypernymy-specific word representations, Yu et al. (2015) propose learning term embeddings by leveraging hypernymy pairs extracted from the web using pattern-based methods. These are then integrated into a dynamic distance-margin model along with "corrupted" negative pairs of words (that are not in a hypernymy relation) with the goal of creating embeddings that capture hypernymy properties. The learned embeddings are further used as features in a supervised learning setting. However, unlike in the current proposal, it is not clear how to evaluate their embeddings in an unsupervised setting. Also, their evaluation seems to be tailored exclusively for detecting hypernymy and is restricted to reasoning about the pairs seen during training time, while the framework in the current proposal could be extended to reason about textual entailment beyond hypernymy and beyond pairs that exist in a taxonomy. Tuan et al. (2016) extend their work by making use of the contextual information between a hyponym and its hypernym. They create term embeddings that are further used as features in

a supervised model to identify new taxonomic relations. It is not trivial how to use the created embeddings in an unsupervised model of hyponymy detection and no explicit feature inclusion is enforced throughout the process of embeddings creation. Furthermore, neither of the two proposals enable determining the directionality of the hypernymy relation given a pair of words for which the relation holds, unlike in the proposed framework.

The most similar previous work, in terms of motivation and aims, is that of Vilnis and McCallum (2015). They also model entailment directly using a vector space, without training a classifier. But instead of representing words as points in a vector space (as in the current approach), they represent words as a Gaussian distribution over points in a vector space. This allows them to represent the extent to which a feature is known versus unknown as the amount of variance in the distribution for that feature's dimension. However the model appears to be more computationally expensive than the current proposal, particularly for inferring vectors. The unsupervised predictions of hyponymy relations are made with the learned vector distributions using Kullback-Leibler divergence between the distributions for the two words. They evaluate their models on the hyponymy data from Baroni et al. (2012). A comparison is provided to their proposed model, but as discussed further in section 2.5.2, the best models in the current proposal achieve non-significantly better average precision than their best models.

Another closely related work is that of Kruszewski et al. (2015). Their semi-supervised model also models entailment in a vector space, but it uses a discrete vector space. They train a mapping from distributional semantic vectors to boolean vectors such that feature inclusion in the boolean space respects a set of entailment relations. They then use feature inclusion to predict hyponymy, and other lexical entailment relations. This approach is similar to the one used in the proposed semi-supervised experiments, except that their discrete entailment prediction operator is very different from the proposed entailment operators in this work. Also, Kruszewski et al. (2015) do not propose any unsupervised method, or any insight into why distributional semantic models extract information that can be useful for entailment. Furthermore, their use of different data for evaluation makes a direct comparison to the current setup unfeasible.

Later, further work has looked at the task of hyponymy detection along with other adjacent related tasks. Nickel and Kiela (2017) propose using hyperbolic embedding spaces to represent hierarchical data from knowledge graphs, and obtain competitive results on graded lexical entailment. Dhingra et al. (2018) further learn word and sentence embeddings in a hyperbolic space that seem to encode hierarchical information that is occurring implicitly in natural language, rather than explicitly in graphs (Nickel and Kiela, 2017).

Nguyen et al. (2017) propose HyperVec, a method to learn hypernymy-specific hierarchical embeddings. The embeddings are created to discriminate hypernymy from other relations, by ensuring that the hypernymy relation is assigned a higher similarity score in the learned embeddings space than other relations. At the same time, the created embeddings enable distinguishing between the hypernym and the hyponym in a given pair of terms, based on the context in which they occur, according to the distributional inclusion hypothesis (Geffet and Dagan, 2005). Vulic and Mrksic (2018) propose a post-processing step to induce linguistic constraints from external resources into a vector space such that the resulting vectors become specialized for entailment. However, their proposal is limited for use on words that appear in such external linguistic resources. This differs from the current proposal in which no external linguistic resource is required to reason about entailment.

Although much of this (later) work seems to follow the direction of encoding entailment in the vector space itself, still none of the above propose entailment operators or inference procedures over already existing word representations as in the proposed framework. Additionally, as it has been previously pointed out (Levy et al., 2015), supervised methods tend to learn properties of one of the words in the pair rather than the relationship between them (also known as the problem of *prototypical hypernyms*), with unsupervised approaches being preferred as they are shown to provide more robust results (Shwartz et al., 2017).

Recently, Chang et al. (2018) model hypernymy based on the distributional inclusion hypothesis (Geffet and Dagan, 2005) according to which the contexts in which a hyponym occurs should be included in the set of contexts in which its hypernym occurs. They obtain word

representations through non-negative matrix factorization of a weighted point-wise mutual information matrix. Feature inclusion is then modeled by ensuring that the created word vectors corresponding to hypernyms have values that are higher or equal than those corresponding to their hyponyms, in every dimension. Unlike most related work, yet similarly to the current proposal, they also propose a set of scoring functions for entailment. However, unlike in the current proposal, no single scoring function performs best across datasets, with the difference in performance being difficult to interpret and not explained theoretically.

In Section 2.5, the experimental setup of Weeds et al. (2014) is replicated, considering both the unsupervised and supervised models as well as their proposed dataset. Weeds et al. (2014) propose an approach to distinguish between different relations that can exist between distributionally similar words by using a supervised approach based on linear SVMs (support vector machine) over vectors encoding pairs of words. They also provide insights into which vector operations seem to be appropriate for detecting different relations: models based on vector addition seem to be providing competitive results on detecting co-hyponymy, while vector differences are being indicative of entailment. Additionally, as further explained in Section 2.5.1, they provide a refinement of a commonly-used dataset in order to prevent supervised models from using artefacts of the data and thus make them comparable in terms of performance to unsupervised approaches. This dataset will be further used for evaluation in Section 2.5.

A comparison is thus provided to the results of the models evaluated by Weeds et al. (2014) and to previously proposed vector-space operators: *cosine*, *dot product* and *vector differences*. The list of considered operators also includes one vector space operator for hyponymy which does not have trained parameters, proposed by Rei and Briscoe (2014), *weighted cosine*. In the case of this directional similarity measure, the dimensions of the dot product (normalised to make it a cosine measure) are weighted such that more weight is placed on the larger values in the entailed (hypernym) vector. This yields an unsupervised model of hyponymy. However, it is not clear how to use such an operator to perform inference, as it is possible within the proposed framework.

2.3 Proposal: modelling entailment in a vector space

In order to develop a model of entailment in a vector space, we consider the definition of entailment as factorised into a conjunction of entailments between binary features. Given two semantic representations y and x, entailment is defined using the logical definition for the vectors of discrete known features: y entails x (further denoted as $y \Rightarrow x$) if and only if all the known features in x are also included in y, that is whenever a feature is known in x, it must also be known in y for the entailment relation to hold. This can be formalized with binary vectors x, y, with 1 representing the known and 0 representing the unknown. The following entailment relations hold: $(1 \Rightarrow 0)$, $(1 \Rightarrow 1)$ and $(0 \Rightarrow 0)$, but $(0 \not\Rightarrow 1)$.

Therefore the definition of entailment should be factorised into features k such that

$$P((y \Rightarrow x) \mid x, y) = \prod_k P((y_k \Rightarrow x_k) \mid x_k, y_k)$$

while the following must hold:

$$P((y_k \Rightarrow x_k) \mid y_k = 0, x_k = 0) = 1$$

$$P((y_k \Rightarrow x_k) \mid y_k = 1, x_k = 0) = 1$$

$$P((y_k \Rightarrow x_k) \mid y_k = 1, x_k = 1) = 1$$

$$P((y_k \Rightarrow x_k) \mid y_k = 0, x_k = 1) = 0$$

with $x_k \in \{0, 1\}$ and $y_k \in \{0, 1\}$.

Then the probability of entailment between two features y_k, x_k can be defined as:

$$P((y_k \Rightarrow x_k) \mid x_k, y_k) = 1 - P(y_k = 0 \mid y_k) P(x_k = 1 \mid x_k)$$

The discrete entailment relation $(y \Rightarrow x)$ can thus be defined with the binary formula:

$$P((y \Rightarrow x) \mid x, y) = \prod_k (1 - (1 - y_k)x_k)$$

Given prior probability distributions $P(x), P(y)$ over these vectors, the exact joint and marginal probabilities for an entailment relation are:

$$P(x, y, (y \Rightarrow x)) = P(x)\ P(y) \prod_k (1 - (1 - y_k)x_k)$$

$$P((y \Rightarrow x)) = E_{P(x)} E_{P(y)} \prod_k (1 - (1 - y_k)x_k) \tag{2.1}$$

Because many important correlations can exist between features and therefore we cannot assume that the features are independent, one should not assume that the priors $P(x)$ and $P(y)$ are factorised. As discussed in Section 2.1, even just representing both a feature f and its negation $\neg f$ requires two different dimensions k and k' in the vector space, because 0 represents unknown and not false. Given valid feature vectors, calculating entailment can consider these two dimensions separately. However the prior $P(x)$ has to enforce the constraint that x_k and $x_{k'}$ are mutually exclusive in order to reason with distributions over vectors. In general, such correlations and anti-correlations exist between many semantic features, which makes inference and calculating the probability of entailment intractable.

To allow for efficient inference in such a model, we propose using a mean-field approximation. This assumes that the posterior distribution over vectors is factorised, but in practice this represents a much weaker assumption than assuming the prior is factorised. The posterior distribution has less uncertainty and therefore is influenced less by non-factorised prior constraints. By assuming a factorised posterior, one can then represent distributions over feature vectors with simple vectors of probabilities of individual features (or as further seen, with their log-odds). These real-valued vectors are the basis of the proposed vector-space model of entailment.

A mean-field model of the discrete space introduced earlier uses vectors μ of real values $\in [0, 1]$

and interprets them as the mean values of the discrete distribution over 0 and 1.

$$\mu_k^y = P(y_k = 1)$$

$$\mu_k^x = P(x_k = 1)$$

In the following, the mean-field approximation for inference of real-valued vectors in entailment graphs is derived. This derivation leads to three proposed vector-space operators for approximating the log-probability of entailment, summarised in Table 2.2. These operators will be used in the evaluation in Section 2.5. This inference framework will also be used in Section 2.4 to model how existing word embeddings can be mapped to vectors to which the entailment operators can be applied.

2.3.1 A mean-field approximation

The mean-field approximation is used to approximate the posterior distribution P using a factorised distribution Q. This provides a concise description of the posterior $P(x|\ldots)$ as a vector of continuous values $Q(x{=}1)$, where $Q(x{=}1)_k = Q(x_k{=}1) \approx E_{P(x|\ldots)}x_k = P(x_k{=}1|\ldots)$ (i.e. the marginal probabilities of each bit). Also, as it will be further shown, this provides efficient methods for doing approximate inference of vectors in a model.

First we consider the simple case where we want to approximate the posterior distribution $P(x, y|(y{\Rightarrow}x))$. In a mean-field approximation, we want to find a factorised distribution $Q(x, y)$ which minimises the Kullback-Leibler (KL) divergence

$$D_{KL}(Q(x, y)||P(x, y|(y{\Rightarrow}x)))$$

with the true distribution $P(x, y|(y{\Rightarrow}x))$. The KL divergence represents a measure of how one probability distribution differs from another probability distribution. Given two probability

distributions Q and P, the KL divergence is defined as

$$\begin{aligned} D_{KL}(Q||P) &= -\sum_x Q(x) \log \frac{P(x)}{Q(x)} \\ &\equiv \sum_x Q(x) \log \frac{Q(x)}{P(x)} \end{aligned}$$

For the particular case at hand, this results in having:

$$\begin{aligned} & D_{KL}(Q(x,y)||P(x,y|(y{\Rightarrow}x))) \\ & \quad = \sum_{xy} Q(x,y) \log \frac{Q(x,y)}{P(x,y|(y{\Rightarrow}x))} \\ & \quad = \sum_{xy} Q(x,y) \log Q(x,y) - \sum_{xy} Q(x,y) \log \frac{P(x,y,(y{\Rightarrow}x))}{P((y{\Rightarrow}x))} \\ & \quad = \sum_{xy} Q(x,y) \log Q(x,y) - \sum_{xy} Q(x,y) \log(\frac{P(x,y) \cdot P((y{\Rightarrow}x)|x,y)}{P((y{\Rightarrow}x))}\ ) \\ & \quad = E_{Q(x,y)} \log(\prod_k Q(x_k) \prod_k Q(y_k)) \\ & \quad - E_{Q(x,y)}(\ \log(\ P(x)P(y)\ ) + \log(\ \prod_k (1{-}(1{-}y_k)x_k)\ ) - \log(\ P((y{\Rightarrow}x))\ )\ ) \end{aligned}$$

Thus we want to minimize

$$\begin{aligned} & min_Q\ \ D_{KL}(Q(x,y)||P(x,y|(y{\Rightarrow}x))) \\ & \qquad = min_Q\ E_{Q(x,y)} \log(\prod_k Q(x_k) \prod_k Q(y_k)) \\ & \qquad - E_{Q(x,y)}(\ \log(\ P(x)P(y)\ ) + \log(\ \prod_k (1{-}(1{-}y_k)x_k)\ )\ ) \end{aligned} \tag{2.2}$$

Which leads to the objective function L

$$\begin{aligned} L\ =\ & \sum_k E_{Q(x_k)} \log Q(x_k) + \sum_k E_{Q(y_k)} \log Q(y_k) \\ & - E_{Q(x)} \log P(x) - E_{Q(y)} \log P(y) - \sum_k E_{Q(x_k)} E_{Q(y_k)} \log(1{-}(1{-}y_k)x_k) \end{aligned}$$

It is to be noted that $(y \Rightarrow x)$ is a variable $\in \{0, 1\}$, depending on the existence of the entailment constraint between any two variables. Thus the prior probability of this constraint, $P((y \Rightarrow x))$, as well as $\log(P((y \Rightarrow x)))$, are not a function of Q, and are dropped in 2.2. In the final equation, the first two terms are the negative entropy of Q, $\sum_k E_{Q(x_k)} \log Q(x_k) + \sum_k E_{Q(y_k)} \log Q(y_k)$, further denoted as $-H(Q)$, which acts as a maximum entropy regulariser, the final term enforces the entailment constraint, and the middle two terms represent the prior for x and y. One approach (generalised further in Section 2.3.2) to the prior terms $E_{Q(x)} \log P(x)$ and $E_{Q(y)} \log P(y)$ is to bound them by assuming $P(x)$ is a function in the exponential family, giving us:

$$\begin{aligned} E_{Q(x)} \log P(x) &\geq E_{Q(x)} \log \frac{\exp(\sum_k \theta_k^x x_k)}{\mathcal{Z}_\theta^x} \\ &= \sum_k E_{Q(x_k)} \theta_k^x x_k - \log \mathcal{Z}_\theta^x \\ &= \sum_k Q(x_k = 1) \theta_k^x - \log \mathcal{Z}_\theta^x \end{aligned}$$

and

$$\begin{aligned} E_{Q(y)} \log P(y) &\geq E_{Q(y)} \log \frac{\exp(\sum_k \theta_k^y y_k)}{\mathcal{Z}_\theta^y} \\ &= \sum_k E_{Q(y_k)} \theta_k^y y_k - \log \mathcal{Z}_\theta^y \\ &= \sum_k Q(y_k = 1) \theta_k^y - \log \mathcal{Z}_\theta^y \end{aligned}$$

where the $\log \mathcal{Z}_\theta$ is not relevant in any of the inference problems and thus will be dropped below. θ_k^x can be seen as the influence of the values taken by the rest of the dimensions $k' \neq k$ on the value at dimension k, thus being a constant with respect to dimension k.

Inference

As typically in mean-field approximations, inference of $Q(x)$ and $Q(y)$ can't be done efficiently with this exact objective L, because of the nonlinear interdependence between x_k and y_k in the

last term. Thus, we introduce two approximations to L, one for use in inferring $Q(x)$ given $Q(y)$ (forward inference), and one for the reverse inference problem (backward inference).

For forward inference we thus have:

$$\begin{aligned}
L \leq & \sum_k (\ E_{Q(x_k)} \log Q(x_k) + E_{Q(y_k)} \log Q(y_k) \\
& - Q(x_k = 1)\theta^x_k - Q(y_k = 1)\theta^y_k \\
& - E_{Q(x_k)} E_{Q(y_k)} \log(1-(1-y_k)x_k)\ ) \\
\approx & \sum_k (\ E_{Q(x_k)} \log Q(x_k) + E_{Q(y_k)} \log Q(y_k) \\
& - Q(x_k = 1)\theta^x_k - Q(y_k = 1)\theta^y_k \\
& - E_{Q(x_k)} \log E_{Q(y_k)} (1-(1-y_k)x_k)\ ) \\
= & - H(Q) - \sum_k (\ Q(x_k{=}1)\theta^x_k - Q(y_k = 1)\theta^y_k \\
& - Q(x_k{=}1) \log Q(y_k{=}1)\ ) \\
:= & L^{F(x)}
\end{aligned} \tag{2.3}$$

where the first inequality corresponds to the application of the bounds and the approximation is obtained due to the discrete nature of x_k and y_k.

We can further optimise this for $Q(x_k{=}1)$:

$$\begin{aligned}
& \frac{\partial L^{F(x)}}{\partial Q(x_k{=}1)} = 0 \\
\equiv\ & \log Q(x_k{=}1) - \log(1{-}Q(x_k{=}1)) - \theta^x_k - \log Q(y_k{=}1) = 0 \\
\equiv\ & \log \frac{Q(x_k{=}1)}{1{-}Q(x_k{=}1)} = +\theta^x_k + \log Q(y_k{=}1) \\
\equiv\ & \log \frac{Q(x_k{=}1)}{Q(x_k{=}0)} = +\theta^x_k + \log Q(y_k{=}1) \\
\equiv\ & \sigma^{\text{-}1}(Q(x_k{=}1)) = +\theta^x_k + \log Q(y_k{=}1) \\
\equiv\ & Q(x_k{=}1) = \sigma(\ \theta^x_k + \log Q(y_k{=}1)\ )
\end{aligned}$$

where $\sigma()$ is the sigmoid function and the first step in the derivation is computing the derivative with respect to u of $u\log(u)$ with $u = Q(x_k = 1)$. the entropy regulariser, making this a specific form of maximum entropy model.

Similarly, for backward inference:

$$\begin{aligned}
L \leq & \sum_k (\ E_{Q(x_k)} \log Q(x_k) + E_{Q(y_k)} \log Q(y_k) \\
& - Q(x_k = 1)\theta_k^x - Q(y_k = 1)\theta_k^y \\
& - E_{Q(x_k)} E_{Q(y_k)} \log(1-(1-y_k)x_k)\ ) \\
\approx & \sum_k (\ E_{Q(x_k)} \log Q(x_k) + E_{Q(y_k)} \log Q(y_k) \\
& - Q(x_k = 1)\theta_k^x - Q(y_k = 1)\theta_k^y \\
& - E_{Q(y_k)} \log E_{Q(x_k)}(1-(1-y_k)x_k)\ ) \\
= & - H(Q) - \sum_k (\ Q(x_k = 1)\theta_k^x - Q(y_k{=}1)\theta_k^y \\
& - (1-Q(y_k{=}1)) \log(1-Q(x_k{=}1))\ ) \\
:= & L^{B(y)}
\end{aligned} \tag{2.4}$$

which we can optimise for $Q(y_k{=}1)$:

$$\begin{aligned}
& \frac{\partial L^{B(y)}}{\partial Q(y_k{=}1)} = 0 \\
\equiv\ & \log Q(y_k{=}1) - \log(1-Q(y_k{=}1)) - \theta_k^y + \log(1-Q(x_k{=}1)) = 0 \\
\equiv\ & \log \frac{Q(y_k{=}1)}{1-Q(y_k{=}1)} = +\theta_k^y - \log(1-Q(x_k{=}1)) \\
\equiv\ & \log \frac{Q(y_k{=}1)}{Q(y_k{=}0)} = +\theta_k^y + \log Q(x_k{=}1) \\
\equiv\ & \sigma^{\text{-}1}(Q(y_k{=}1)) = +\theta_k^y + \log Q(x_k{=}1) \\
\equiv & Q(y_k{=}1) = \ \sigma(\ \theta_k^y - \log(1-Q(x_k{=}1))\ )
\end{aligned} \tag{2.5}$$

Entailment operators

Note that in equations (2.3) and (2.4) the final terms, $Q(x_k{=}1)\log Q(y_k{=}1)$ and $(1{-}Q(y_k{=}1))\log(1{-}Q(x_k{=}1))$ respectively, are approximations to the log-probability of the entailment. We define two vector-space operators, $\circledcirc$ and $\circledgtr$, to be these same approximations. This is obtained using Jensen's inequality, defined as $\varphi(E[X]) \leq E[\varphi(X)]$ for any convex function φ, with the inequality being reversed in the case of concave functions.

$$\begin{aligned}
\log Q(y{\Rightarrow}x) &= \log(E_{Q(x)}E_{Q(y)}\prod_k(1-(1{-}y_k)x_k)) \\
&= \sum_k \log(E_{Q(x_k)}E_{Q(y_k)}(1-(1{-}y_k)x_k)) \\
&\geq \sum_k E_{Q(x_k)}\log(E_{Q(y_k)}(1-(1{-}y_k)x_k)) \\
&= \sum_k Q(x_k{=}1)\log Q(y_k{=}1) \\
&= Q(x{=}1)\cdot\log Q(y{=}1) \\
&\equiv X\circledcirc Y
\end{aligned}$$

$$\begin{aligned}
\log Q(y{\Rightarrow}x) &= \log(E_{Q(x)}E_{Q(y)}\prod_k(1-(1{-}y_k)x_k)) \\
&= \sum_k \log(E_{Q(x_k)}E_{Q(y_k)}(1-(1{-}y_k)x_k)) \\
&\geq \sum_k E_{Q(y_k)}\log(E_{Q(x_k)}(1-(1{-}y_k)x_k)) \\
&= \sum_k (1{-}Q(y_k{=}1))\log(1{-}Q(x_k{=}1)) \\
&= (1{-}Q(y{=}1))\cdot\log(1{-}Q(x{=}1)) \\
&\equiv Y\circledgtr X
\end{aligned}$$

We parametrise these operators with the vectors X, Y of log-odds of $Q(x), Q(y)$, namely $X = \log\frac{Q(x=1)}{Q(x=0)} = \sigma^{-1}(Q(x{=}1))$. The resulting operator definitions are summarised in Table 2.2.

$$
\begin{aligned}
X \circledcirc Y &\equiv \sigma(X) \cdot \log \sigma(Y) \\
Y \circledcirc X &\equiv \sigma(-Y) \cdot \log \sigma(-X) \\
Y \tilde{\Rightarrow} X &\equiv \sum_k \log(1 - \sigma(-Y_k)\sigma(X_k))
\end{aligned}
$$

Table 2.2: The proposed entailment operators, approximating $\log P(y \Rightarrow x)$.

Also the probability of entailment given in equation (2.1) becomes factorised when we replace P with Q. We define a third vector-space operator, $\tilde{\Rightarrow}$, to be this factorised approximation, also shown in Table 2.2.

2.3.2 Extension to entailment graphs

In general, doing inference for one entailment is not enough; we want to be able to do inference in a graph of entailments between variables. In this section we generalise the above mean-field approximation to inference of variables in entailment graphs.

To represent information about variables that comes from outside the entailment graph, we assume we are given a prior $P(x)$ over all variables x_i in the graph. As previously, we do not assume that the prior $P(x)$ is factorised, but that it can be approximated with a mean-field approximation.

Given a set of variables x_i each representing vectors of binary variables x_{ik}, a set of entailment relations $r = \{(i,j)|(x_i \Rightarrow x_j)\}$, and a set of negated entailment relations $\bar{r} = \{(i,j)|(x_i \not\Rightarrow x_j)\}$, we can write the joint posterior probability as:

$$
\begin{aligned}
P(x, r, \bar{r}) = \frac{1}{\mathcal{Z}} P(x) \prod_i \Big(& \\
& \Big(\prod_{j:r(i,j)} \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk}) \Big) \\
& \Big(\prod_{j:\bar{r}(i,j)} (1 - \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk})) \Big) \Big)
\end{aligned}
$$

The goal to find a factorised distribution Q that minimises $L = D_{KL}(Q(x)||P(x|r,\bar{r}))$. As

previously, we bound this loss for each element $X_{ik}{=}\,\sigma^{\text{-}1}(Q(x_{ik}{=}1))$ of each vector we want to infer, using analogous Jensen's inequalities for the terms involving nodes i and j such that $r(i,j)$ or $r(j,i)$. For completeness, we also propose similar inequalities for nodes i and j such that $\bar{r}(i,j)$ or $\bar{r}(j,i)$, and bound them using the constants C_{ijk}. If entailment cannot be established based on any of the dimensions $k' \neq k$, then one no longer needs to look at dimension k (as the lack of entailment is already decided by the non-entailing dimensions).

$$C_{ijk} \leq \prod_{k' \neq k} (1{-}(1{-}E_{Q(x_{ik'})}x_{ik'})E_{Q(x_{jk'})}x_{jk'}) = \prod_{k' \neq k} (1{-}\sigma(-X_{ik'})\sigma(X_{jk'})).$$

As previously, to represent the prior $P(x)$, we use the terms θ_{ik}.

$$\theta_{ik} \geq \log \frac{E_{Q(x_{\bar{i}\bar{k}})}P(x_{\bar{i}\bar{k}}, x_{ik}{=}1)}{1 - E_{Q(x_{\bar{i}\bar{k}})}P(x_{\bar{i}\bar{k}}, x_{ik}{=}1)}$$

where $x_{\bar{i}\bar{k}}$ is the set of all $x_{i'k'}$ such that either $i'{\neq}i$ or $k'{\neq}k$. These terms can be thought of as the log-odds terms that would be contributed to the loss function by including the prior's graphical model in the mean-field approximation.

Now we can infer the optimal X_{ik} as:

$$\begin{aligned} X_{ik} \;=\; \theta_{ik} \;+ \sum_{j:r(i,j)} & -\log\sigma(-X_{jk}) \\ + \sum_{j:r(j,i)} \log\sigma(X_{jk}) \;+ \sum_{j:\bar{r}(j,i)} & \log\frac{1{-}C_{ijk}\sigma(X_{jk})}{1{-}C_{ijk}} \\ + \sum_{j:\bar{r}(i,j)} & -\log\frac{1{-}C_{ijk}\sigma(-X_{jk})}{1{-}C_{ijk}} \end{aligned}$$

In summary, the proposed mean-field approximation does inference in entailment graphs by iteratively re-estimating each X_i as the sum of: the prior log-odds, $-\log\sigma(-X_j)$ for each entailed variable j, and $\log\sigma(X_j)$ for each entailing variable j.[1] This inference optimises $X_i{\circledcirc}X_j$

[1] It is interesting to note that $-\log\sigma(-X_j)$ is a non-negative transform of X_j, similar to the ReLU nonlinearity which is popular in deep neural networks (Glorot et al., 2011). This transform reflects the fact that it is only the features known in x that influence the possible y. $\log\sigma(X_j)$ is the analogous non-positive transform.

for each entailing j plus $X_i \circledcirc X_j$ for each entailed j, plus a maximum entropy regulariser on X_i. Negative entailment relations, if they exist, can also be incorporated with some additional approximations.

2.4 Interpreting distributional semantics vectors

In order to evaluate how well the proposed framework in Section 2.3 provides a formal foundation for the distributional semantics of entailment, we use it to re-interpret an existing model of distributional semantics in terms of semantic entailment.

There has been a lot of work on how to use the distribution of contexts in which a word occurs to induce a vector representation that reflects the semantics of the word. Thus distributional semantics learns the semantics of words by looking at the distribution of contexts in which they occur. We could therefore leverage previous work on distributional semantics by mapping the produced word representations to vectors in the proposed framework. This implies re-interpreting an existing distributional semantics model and further using the proposed entailment operators to predict entailment between words using these vectors.

An evaluation of such predictions will be provided in Section 2.5 on the task of hyponymy detection. In the following, three different options are provided for the interpretation of an existing distributional semantics model: Word2Vec (Mikolov et al., 2013a,b) as an approximation to an entailment-based model of the semantic relationship between a word and its context.

To model the relationship between words and their contexts, we assume that the semantic features of a word (further referred to as the middle word) are (statistically speaking) redundant with those of its context words, and consistent with those of its context words. One can thus consider a hidden vector which is the consistent unification of the features of the word and its context. In other words, we assume the existence of a hidden vector which entails both of the (middle) word representation and the context word representations, and which is consistent

with prior constraints on these vector representations. The process can be split into two steps: inferring the hidden vector Y from the middle vector X_m, context vectors X_c and prior, and then computing the log-probability (2.6) that this hidden vector entails the middle and context vectors:

$$\max_Y(\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c)) \tag{2.6}$$

One possible interpretation would be that Word2Vec's Skip-Gram model is learning its context and middle word vectors such that the log-probability of this entailment is high for the observed context words and low for other (sampled) context words. The word embeddings produced by Word2Vec within the Skip-gram model are thus related to the vectors X_m assigned to the middle words. In a similar fashion, the vectors X_c represent the Word2Vec vectors of the context words. We denote by X'_c the context vectors of Word2Vec that combine (as in equation (2.5)) information about a context word itself with information which can be inferred from this word given the prior, $X'_c = \theta_c - \log\sigma(-X_c)$. Further, the Word2Vec vectors can be interpreted as the log-odds of features being known. In this case we can treat these vectors directly as the X_m in the model.

The inferred hidden vector Y can then be calculated using the model of backward inference from Section 2.3, taking into account that both the context vector and the middle vector are needed to infer the hidden vector Y as discussed previously.

$$\begin{aligned} Y &= \theta_c - \log\sigma(-X_c) - \log\sigma(-X_m) \\ &= X'_c - \log\sigma(-X_m) \end{aligned}$$

Since the unification Y of context and middle word features is computed using backward inference, we also use the backward-inference operator $\oslash$ to calculate how successful that unification was. This is the *Log-odds* interpretation and the final score obtained is:

$$
\begin{aligned}
&\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\
&\quad = Y{\circledcirc}X_m + Y{\circledcirc}X_c + Q(y=1){\cdot}\theta_c \\
&\quad = Y{\circledcirc}X_m + Y{\circledcirc}X_c + (\sigma(Y) \cdot \theta_c) \\
&\quad \approx Y{\circledcirc}X_m + Y{\circledcirc}X_c + (-\sigma(-Y){\cdot}\theta_c) \\
&\quad = Y{\circledcirc}X_m + \sigma(-Y){\cdot}\log\sigma(-X_c) - \sigma(-Y){\cdot}\theta_c \\
&\quad = Y{\circledcirc}X_m - \sigma(-Y){\cdot}X'_c
\end{aligned}
$$

The second equality follows from the first based on having $Y = \log\frac{Q(y=1)}{Q(y=0)} = \sigma^{-1}(Q(y{=}1))$ and thus $\sigma(Y) = Q(y=1)$ as in 2.5. This interpretation ignores however the equivalence in Word2Vec between pairs of positive values and pairs of negative values, due to its use of the dot product. As a more accurate interpretation, we interpret each Word2Vec dimension as specifying whether its feature is known to be true or known to be false. Translating this Word2Vec vector into a vector in the proposed entailment vector space, we get one copy Y^+ of the vector representing known-to-be-true features and a second negated duplicate Y^- of the vector representing known-to-be-false features, which we can concatenate to get our representation Y. This is further referred to as the *Dup* interpretation. We thus obtain:

$$
\begin{aligned}
Y^+ &= X'_c - \log\sigma(-X_m) \\
Y^- &= -X'_c - \log\sigma(X_m)
\end{aligned}
$$

Then by replacing Y^+ and Y^- into the previous formulation, we obtain the log probability of entailment as:

$$
\begin{aligned}
&\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\
&\quad \approx Y^+{\circledcirc}X_m + (-\sigma(-Y^+)){\cdot}X'_c + Y^-{\circledcirc}(-X_m) + (-\sigma(-Y^-)){\cdot}(-X'_c)
\end{aligned}
$$

A third alternative is to add some probability mass reserved for unknown in the vicinity of zero. By subtracting 1 from both the original and negated copies of each dimension, we get a

probability of unknown of $1-\sigma(X_m-1) - \sigma(-X_m-1)$, namely the probability left over after adding the probabilities of the original feature and its negation. This is referred to as the *Unk dup* interpretation.

This gives us:

$$Y^+ = X'_c - \log\sigma(-(X_m-1))$$
$$Y^- = -X'_c - \log\sigma(-(-X_m-1))$$

Then the log probability of entailment is:

$$\log P(y, y\Rightarrow x_m, y\Rightarrow x_c)$$
$$\approx Y^+\circledcirc(X_m-1) + (-\sigma(-Y^+))\cdot X'_c + Y^-\circledcirc(-X_m-1)) + (-\sigma(-Y^-))\cdot(-X'_c)$$

To understand better the relative accuracy of these three interpretations, Figure 2.1 shows a comparison of the training gradient which Word2Vec uses to train its middle-word vectors to the training gradient for each of these interpretations. The gradients are plotted for the range of values typically found in Word2Vec vectors for both the middle vector and the context vector. As expected, the second interpretation is more accurate than the first (referred to as Log-odds further) because its plot is anti-symmetric around the diagonal, like the Word2Vec gradient. In the third alternative (Unk dup), the constant 1 was chosen to optimise this match, rather than assuming no uncertainty (a constant of 0). This produced a close match to the Word2Vec training gradient, as shown in Figure 2.1 (*Word2Vec* versus *Unk dup*).

Thus, Word2Vec can be seen as a good approximation to the third model, and a progressively worse approximation to the second and first models. Section 2.5 further evaluates the three interpretations on the task of hyponymy detection.

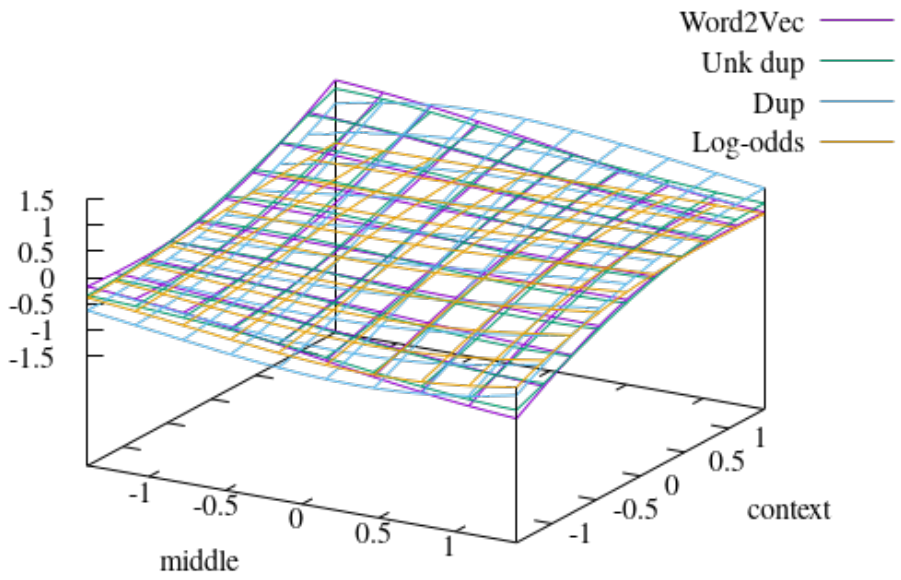


Figure 2.1: The learning gradients for Word2Vec, the *log-odds* $\circledcirc$, the *dup* $\circledcirc$ and the *unk dup* $\circledcirc$ interpretation of its vectors.

2.5 Evaluation and results

In the following the quality of the interpretations provided in Section 2.4 and further the performance of the entailment operators proposed in Section 2.3 will be evaluated.

It is also important to note that determining whether entailment holds between two sentences and in which direction depends on the ability to identify hyponyms (Weeds et al., 2014). It is therefore natural to first investigate the proposed framework and operators on the task of hyponymy detection. In this sense, one approach is to consider that most of the semantic features of a hyponym (e.g. “dog”) must be included in the semantic features of the hypernym (e.g. “animal”), but not necessarily in the opposite direction. Translated in the current framework, that would imply that whenever a feature is “known” for the hyponym (e.g. “dog”), it must also be “known” for its hypernym (e.g. “animal”), i.e. whatever a “dog” might represent should also be represented by an “animal”.

2.5.1 Experimental setup

A variant of the BLESS dataset (Baroni and Lenci, 2011) is considered for evaluation as it represents the benchmark for the given task. In its original form, BLESS represents a collection of noun-noun pairs constructed on the basis of 200 concrete monosemous nouns, denoted as BLESS concepts, further equally divided between living and non-living entities, and grouped into 17 broader classes (e.g., bird, furniture, vehicle, etc.). Each pair denotes one of the following relations: hyponymy (denoting a superset relationship like for example "animal" - "dog"), co-hyponymy (denoting words that share a common hypernym like in "dog" - "cat"), meronymy (denoting a part-whole relationship like "tail" - "dog") or random (denoting no relationship between the words like in "dog" - "computer"). The modified version (Weeds et al., 2014) refines the dataset to fulfill some pre-established conditions in order to prevent classifiers to make use of artefacts of the data. Such conditions ensure (among others) that just considering the distributional similarity of words is not enough to do well on the task.

For evaluation on hyponymy detection, the experimental setup of Weeds et al. (2014) is replicated, using the provided selection of word pairs [2] from the BLESS dataset. Contrary to the approach in Weeds et al. (2014), publicly available pretrained 300-dimensional Word2Vec word embeddings [3] were considered, as they had shown competitive results on a series of tasks. Out of the 1667 word pairs in the original dataset, 24 were removed due to the lack of embeddings for either of the words in the pair. The considered noun-noun word pairs include positive hyponymy pairs as well as negative pairs. The negative pairs are formed either by reversing the hyponymy pairs or by considering pairs of words in other semantic relationships than hyponymy (co-hyponymy, meronymy or no relation). Their selection is balanced between positive and negative examples, so that accuracy can be used as the performance measure.

Two experimental setups are considered: semi-supervised and unsupervised. To evaluate the proposed entailment operators in the semi-supervised setup, a linear mapping is trained into a

[2] https://github.com/SussexCompSem/learninghypernyms
[3] https://code.google.com/archive/p/word2vec/

new vector space in which the entailment operators are applied to predict hyponymy. For the semi-supervised experiments, ten-fold cross validation is used, where for each test set, items are removed from the associated training set if they contain any word from the test set. Thus, the vocabulary of the training and testing sets are always disjoint, thereby requiring that the models learn about the vector space and not about the words themselves.

2.5.2 Results and discussion

The hyponymy detection results are given in Table 2.3 for the unsupervised approach and in Table 2.4 for the semi-supervised one. Two measures of performance are reported: hyponymy detection accuracy (*50% Acc*) and direction classification accuracy (*Dir Acc*).

Since all the operators only determine a score, a threshold was required for the hyponymy detection accuracy. Given that the proportion of positive examples in the dataset has been artificially set to 50%, the threshold was set at the point where the proportion of positive examples output is 50%, further called "*50% Acc*". Thus the threshold is set after seeing the testing inputs but not their target labels.

Direction classification accuracy (*Dir Acc*) indicates how well the method distinguishes the relative abstractness of two nouns. Given a pair of nouns which are in a hyponymy relation, the goal is to determine which word is the hypernym and which is the hyponym. The "Dir Acc" measure only considers positive examples and chooses one of two directions, so it is inherently a balanced binary classification task. Classification is performed by simply comparing the scores in both directions and considering the highest one. If both directions produce the same score, the expected random accuracy (50%) is used.

The best results from Weeds et al. (2014) are reported as representative of previous work. They use the same testing methodology and hyponymy data as in the current work, with the only difference being the word embeddings as explained previously. For the semi-supervised models, Weeds et al. (2014) train SVM classifiers, which are potentially more powerful than the

proposed linear vector mappings.

Unsupervised Hyponymy Detection

A first set of experiments evaluate the vector-space operators in unsupervised models of hyponymy detection. As argued in Section 2.4, the three progressively more accurate interpretations of Word2Vec vectors are evaluated in the proposed framework: the log-odds interpretation using the backward-inference entailment operator (*log-odds* $\oslash$), the negated duplicate interpretation (*dup* $\oslash$), and the negated duplicate interpretation with unknown around zero (*unk dup* $\oslash$). Additionally, the factorised calculation of entailment is also used with the log-odds interpretation (*log-odds* $\vec{\Rightarrow}$), as well as (*dup* $\vec{\Rightarrow}$) and (*unk dup* $\vec{\Rightarrow}$), and the forward-inference entailment operator (*log-odds* $\oslash$), neither of which match the proposed interpretations.

Further, the proposed operators are also compared to the dot product (*dot*), vector differences (*diff*), cosine (*cos*) and the weighted cosine proposed by Rei and Briscoe (2014) (*weighted cos*), all computed using the same word embeddings as for the proposed operators.

The comparison to the dot product (*dot*) is explained by the fact that it represents the standard vector-space operator and has been shown to capture semantic similarity very well. However, because the dot product is a symmetric operator, it always performs at chance for direction classification as seen from Table 2.3. Another vector-space operator which has received much attention is vector differences (*diff*). This is used (with vector sum) to perform semantic transforms, such as "king - male + female = queen", and has previously been used for modelling hyponymy (Vylomova et al., 2015; Weeds et al., 2014). In the current work, the pairwise differences are summed to get a score which is further used for hyponymy detection.

For the unsupervised results in Table 2.3, the best unsupervised model of Weeds et al. (2014), and the operators *cos*, *dot*, *diff* and *weighted cos* all perform similarly on accuracy, as does the log-odds factorised entailment calculation (*log-odds* $\vec{\Rightarrow}$). The log-odds forward-inference entailment operator (*log-odds* $\oslash$) performs above chance but not well, as expected given the

operator	50% Acc	Dir Acc
(Weeds et al., 2014)	58%	–
log-odds $\oslash$	54.0%	55.9%
weighted cos	55.5%	57.9%
dup $\vec{\Rightarrow}$	55.6%	50.5%
cos	55.9%	51.8%
dot	56.3%	50%
diff	56.9%	59.6%
log-odds $\vec{\Rightarrow}$	57.0%	59.4%
log-odds $\oslash$	60.1%*	62.2%
dup $\oslash$	61.7%	68.8%
unk dup $\vec{\Rightarrow}$	63.4%*	68.8%
unk dup $\oslash$	64.5%	68.8%

Table 2.3: Accuracies on the BLESS data from Weeds et al. (2014),using the Google-News word embeddings for hyponymy detection (*50% Acc*) and hyponymy direction classification (*Dir Acc*) in the unsupervised experiments. * marks a significant difference with the previous row.

backward-inference-based interpretation of Word2Vec vectors. By definition, *dot* is at chance for direction classification, but the other models all perform better, indicating that all these operators are able to measure relative abstractness. As predicted, the backward-inference entailment operator $\oslash$ performs significantly better than all the other operators on accuracy, as well as on direction classification, even assuming the log-odds interpretation of Word2Vec vectors.

When moving further to the more accurate interpretation of Word2Vec vectors as specifying both original and negated features (*dup* $\oslash$), improvements (non-significant) can be observed over the log-odds interpretation *log-odds* $\oslash$. Finally, the third and most accurate interpretation, where values around zero can be unknown (*unk dup* $\oslash$), achieves the best results in unsupervised hyponymy detection, as well as for direction classification. Changing to the factorised entailment operator (*unk dup* $\vec{\Rightarrow}$) is worse than the *unk dup* $\oslash$, but still significantly better than the results obtained using the other operators.

To allow a direct comparison to the model of Vilnis and McCallum (2015), an additional evaluation of the the unsupervised models was performed on the hyponymy data from Baroni et al. (2012). The best model achieved 81% average precision on this dataset, non-significantly better

than the 80% achieved by the best model reported by Vilnis and McCallum (2015).

Semi-supervised Hyponymy Detection

Since the unsupervised learning of word embeddings may reflect many context-word correlations which have nothing to do with hyponymy, a semi-supervised setting was also considered. Adding some supervision helps distinguish features that capture semantic properties from other features which are not relevant to hyponymy detection. But even with supervision, the goal is to have the resulting model captured in a vector space, and not in a parametrised scoring function. To achieve that, mappings are trained from the Word2Vec word vectors to new word vectors. Further, the entailment operators are applied in the new vector space to predict hyponymy. Because the words in the testing set are always disjoint from the words in the training set, this experiment measures how well the original unsupervised vector space captures features that generalise entailment across words, and not how well the mapping can learn about individual words.

The objective is to learn a mapping to a new vector space in which an operator can be applied to predict hyponymy. The linear mappings are trained for the $\circ$ operator (*mapped* $\circ$) as it represents the best performing proposed operator in the unsupervised experiments and for vector differences (*mapped diff*), representing the baseline operator. Additionally, two sets of experiments consider the forward-inference entailment operator (*mapped* $\circledcirc$) and the factorised entailment calculation (*mapped* $\Rightarrow$). The duplicated interpretations (*dup* $\circ$, *unk dup* $\circ$) are not used because these transforms are subsumed by the ability to learn a linear mapping. Previous work on using vector differences for semi-supervised hyponymy detection has used a linear SVM (Vylomova et al., 2015; Weeds et al., 2014).However, a cross entropy loss is used in the current work, while previous work uses a large-margin loss and SVM training.

The semi-supervised results in Table 2.4 show a similar pattern to the unsupervised results in Table 2.4. The $\circ$ operator achieves the best generalization from training word vectors to testing word vectors. The *mapped* $\circ$ model has the best accuracy, followed by the factorised entailment

operator	supervision	50% Acc	Dir Acc
(Weeds et al., 2014)	SVM	75%	–
mapped diff	cross ent	64.3%	72.3%
mapped $\oslash$	cross ent	74.5%	91.0%
mapped $\tilde{\Rightarrow}$	cross ent	77.5%	92.3%
mapped $\oslash$	cross ent	80.1%	90.0%

Table 2.4: Accuracies on the BLESS data from Weeds et al. (2014), using the Google-News word embeddings for hyponymy detection (*50% Acc*) and hyponymy direction classification (*Dir Acc*), in the semi-supervised experiments.

operator *mapped* $\tilde{\Rightarrow}$ and Weeds et al. (2014). Direction accuracies of all the proposed operators (*mapped* $\oslash$, *mapped* $\tilde{\Rightarrow}$, *mapped* $\oslash$) reach into the 90's, much better than with vector differences (*mapped diff*). The *diff* operator performs particularly poorly in this *mapped* setting, perhaps because both the mapping and the operator are linear. These semi-supervised results again support the distributional-semantic interpretations of Word2Vec vectors and their associated entailment operator $\oslash$.

2.6 Conclusion

With recognizing textual entailment being an important aspect in natural language understanding tasks, the idea adopted in the current proposal is to use specialized vector space models to encode various lexical relations such as hypernymy. This is motivated by the fact that existing distributional semantics creates vector-space representations that effectively capture many forms of semantic similarity, yet their relation to semantic entailment has been less clear (Levy et al., 2015; Henderson and Popa, 2016).

A vector-space model is thus proposed to provide a formal foundation for a distributional semantics of entailment (Henderson and Popa, 2016). Using a mean-field approximation to logical entailment of binary vectors, we develop approximate inference procedures and entailment operators over vectors of probabilities of features being known (versus unknown). This framework is also used to reinterpret an existing distributional-semantic model (Word2Vec) as approximat-

ing an entailment-based model of the distributions of words in contexts, thereby predicting lexical entailment relations. Three increasingly accurate approximations are provided with more accurate interpretations resulting in more accurate unsupervised models of lexical entailment.

In both unsupervised and semi-supervised experiments on hyponymy detection, substantial improvements are obtained over previous results. Additionally, the proposed model yields vectors that obtain competitive results on direction classification as well, indicating that the method enables distinguishing the relative abstractness of two nouns. A crucial distinction between the semi-supervised models proposed here and much previous work is that they learn a mapping into a vector space which represents entailment, rather than learning a parametrised entailment classifier. Within this new vector space, the entailment operators and inference equations apply, thereby generalising naturally from these lexical representations to the compositional semantics of multi-word expressions and sentences.

Further work is needed to explore the full power of these abilities to extract information about entailment from both unlabelled text and labelled entailment data, encode it all in a single vector space, and efficiently perform complex inferences about vectors and entailments. This future work on compositional distributional semantics should further demonstrate the full power of the proposed framework for modelling entailment in a vector space.

Chapter 3

Unsupervised syntax-aware contextualized word representations

In an effort to account for language variability and the insufficiency of generic word type embeddings to account for all senses and possible usages of a word, some work has looked at methods to improve the representation of words based on their individual use within a particular context. While some work focused on modelling the context around a target word to further obtain an enriched representation of the word itself (Melamud et al., 2016), other focused directly on the creation of a context-aware representation for the word, also denoted as *(word) token embedding* or *token-based representation* (Tu et al., 2017; Popa et al., 2019a,b). The idea behind token embeddings is to represent a word in its context, with the same word bearing different representations in different contexts. This contrasts to the generic word type embedding representation, which is the same in every context. Furthermore, as contextuality and compositionality are interlinked, and provided the structure of the sentence is taken into account when creating such token representations, such an approach could lead to including some notion of compositionality within the word representations themselves rather than relying only on the parameters of an architecture built on top of them.

In the following, the relevant literature around context-aware word representations is presented

in Section 3.1 in comparison to the current proposal. Section 3.2 details the proposal for the computation of token embeddings, while Section 3.3 provides the experimental protocol followed for the evaluation of the proposed token embeddings. Results of using token embeddings on a series of commonly considered sentence understanding tasks are presented and discussed extensively in Section 3.4, along with quantitative and qualitative analysis. Additionally, a study on the application of contextualized representations to the task of implicit discourse relation classification is provided in Section 3.5 as also outlined in Popa et al. (2019a,b). Finally Section 3.6 summarizes the contribution of the current proposal and discusses possible future directions.

3.1 Related work

To obtain context-sensitive word embeddings, most previous work has treated this as a word sense disambiguation problem, with each word representing a collection of discrete mutually exclusive senses, individually represented by different vectors - typically one vector per word sense (Neelakantan et al., 2014; Chen et al., 2014; Guo et al., 2014; Liu et al., 2015). In these approaches there is a fixed number of vectors to learn for each word which is based on the number of senses a word can have. Dasigi et al. (2017) also propose a method to obtain context-aware token embeddings, however they produce their embeddings by estimating a distribution over semantic concepts (synsets) extracted from WordNet (Miller, 1995). They test the proposed embeddings to predict prepositional phrase attachments. Kawakami and Dyer (2015) use cross-lingual supervision and bidirectional Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) to provide context-sensitive word representations. Their method is based on the intuition that a contextualized representation of a word is sufficiently accurate if it enables the selection of the correct translation of that word in another language. Their obtained representations are tested on the tasks of lexical substitution, low resource machine translation and prediction of semantic supersenses.

In contrast to these methods, the current proposal uses neither lexical resources nor multilingual data and does not rely on the existance of a pre-defined list of senses for a word. Instead, it leverages syntactic information from parse trees to derive a different representation for each occurrence of a word, namely each word token (Popa et al., 2019a,b).

Melamud et al. (2016) propose *context2vec*, a model for learning a generic embedding function for the contexts around a target word. Their proposal is based on the Continuous Bag-of-Words (CBOW) (Mikolov et al., 2013a) architecture, yet provides a different way to embed the context: instead of representing the context of a target word by an average of the word embeddings of its context words, Melamud et al. (2016) propose using a variant of bidirectional LSTM networks. Thus the local context of a target word is considered by running two LSTMs on its right and left vicinity with the goal of having the context predict the target through a log linear model. The parameters of the two LSTMs are separate and are intended for use on the left (from left-to-right) and right (from right-to-left) contexts respectively. The two obtained representations are further concatenated and fed as input to a multi-layer perceptron, whose output is considered to be the context embedding. Finally, the objective is to embed the target word and its context in the same space such that dependencies between them are preserved. The method is evaluated on the tasks of sentence completion, word sense disambiguation and lexical substitution. Although *context2vec* focuses on modelling the context alone and is not targeted at modelling the meaning of words in context, authors note this could potentially be achieved as well. Nevertheless, the method proposed does not use syntactic information and does not provide one different representation for each word token. Moreover, the obtained representations are not tested on tasks that require reasoning beyond word level such as sentence understanding tasks.

The method proposed in the current thesis provides a different word token vector for every individual context and not just for each individual sense of a word and does not limit itself to the modelling of contexts. In this respect, one proposal related to the current work is that of Tu et al. (2017), who suggest learning embeddings for tokens in an unsupervised manner, by leveraging information from the local context of each token, using a standard auto-encoder architecture.

Their learned token embeddings represent transformations of the type embeddings they denote along with transformations of the type embeddings their neighbouring tokens denote. They provide two different approaches to compute representations for tokens: either through feed-forward encoders or using recurrent neural network encoders.

Given a sequence of n tokens $t = [t_1, t_2, \ldots, t_n]$, let x_k denote the word type that corresponds to token t_k, let $e_{x_k} \in \mathbb{R}^d$ denote the pre-trained embedding of the x_k word type (which is assumed to exist) and let w' be the window size considered such that it contains: the token t_k, w' tokens before it and w' tokens after it. Let f be a set of functions that take as input a sequence of tokens t and the index k of a token of interest and provide as output a d'-dimensional vector representing the embedding of the sequence of tokens contained in the window of text surrounding the k^{th} token:

$$f(t, k) = q(W^{(D)} v_k + b^{(D)})$$

$$v_k = [e_{x_{(k-w')}}; e_{x_{(k-w')+1}}; \ldots; e_{x_k}; \ldots; e_{x_{(k+w')-1}}; e_{x_{(k+w')}}]$$

where q stands for an element-wise nonlinear function (e.g. tanh), $W^{(D)} \in \mathbb{R}^{d' \times d(2w'+1)}$ is a matrix of parameters, (;) stands for vertical concatenation and $b^{(D)} \in \mathbb{R}^{d'}$ is a bias vector. Training is performed using the autoencoder architecture with the feed-forward encoder f and an analogous fully-connected layer decoder g : f converts the input to a vector and further g attempts to reconstruct the input from that vector. The training objective is minimising a weighted squared difference between the input and the reconstructed input. The weighting scheme is adopted in order to account for the focus on a particular position in the encoded sequence that corresponds to the token of interest.

$$loss_{WRE}(f, g, t, k) = \sum_{i=1}^{|t|} w_i \left\| g(f(t, k))_i - e_{t_i} \right\|_2^2$$

where w_i is the weight of reconstructing the i'th token and $g(f(t, k))_i$ is the subvector of $g(f(t, k))$ corresponding to the i'th entry in the sequence. Tu et al. (2017) report best results are obtained when using a "focused" weighting scheme, with $w_j = 2$ and $w_i = 1$ for $i \neq j$.

Alternatively, considering the same scheme of encoding a token along with its context from a given window size, Tu et al. (2017) note that one can use a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) encoder. In this case, the sequence containing the token t_k is encoded and the final hidden vector h_k stands for the d'-dimensional encoding of the token. The training is done with a sequence-to-sequence "seq2seq" autoencoder (Sutskever et al., 2014) that uses one LSTM as encoder f and another LSTM as the decoder g, initialized with the output of f.

The produced token embeddings are further evaluated as features for downstream low-resource tasks like Twitter part-of-speech tagging and dependency parsing. Regardless of the choice of encoder, when input to the downstream tasks, the embeddings of the tokens are concatenated to their corresponding word type embeddings and to a binary feature vector. Unfortunately, that makes it difficult to determine to what extent it is the token embeddings that are responsible for the reported scores and which are the individual contributions of the pre-trained word embeddings and of the feature vectors.

It is also important to note that, although they produce token embeddings in an unsupervised manner, Tu et al. (2017) do not leverage in any way the sentence parse tree to induce syntactic information to the token embeddings, limiting themselves to the local context (as defined by a window of words) to estimate the token representations. It is thus not straightforward how to explicitly add the syntactic information on top of their proposed model. Moreover, it is not clear how these representations behave in a sentence understanding scenario. In such a setup long distance dependencies could account for more than in the considered tasks and an encoding of longer sequences (maybe entire sentences) might be preferred as opposed to the considered tasks, where Tu et al. (2017) did not find large context windows to be helpful.

In computer vision it has become standard practice to re-use weights of deep convolutional neural networks (CNNs) pre-trained on large supervised training sets like the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset (Deng et al., 2009) to initialize multiple deep layers for a series of tasks (Socher et al., 2014; Gao et al., 2015; Gidaris and Komodakis,

2015; Shelhamer et al., 2017). On the other hand, in natural language processing most work uses word embeddings pre-trained on large amounts of data to initialize only the lowest layer of deep learning models, namely the word vectors. Inspired by this, McCann et al. (2017) propose a way to obtain context-aware word vectors (CoVe) by leveraging a deep LSTM encoder. More precisely, they transfer the weights of a pre-trained deep LSTM encoder, coming from a sequence-to-sequence model trained for machine translation (MT-LSTM), to a variety of NLP tasks. These tasks include: question classification, textual entailment recognition, question answering and sentiment analysis. They show that the produced contextualized representations constitute improvements on these tasks when used in combination with standard pre-trained word embeddings and character vectors.

The translation model McCann et al. (2017) use is an attention-based sequence-to-sequence model for English-to-German translation. It follows a standard encoder-decoder architecture as explained further: given an input sequence of n words in a source language

$$w^x = [w_1^x, w_2^x, \ldots, w_n^x]$$

and aiming to produce a sequence of m words in a target language

$$w^y = [w_1^y, w_2^y, \ldots, w_m^y]$$

, GloVe(w^x) denotes the sequence of GloVe vectors that correspond to the words in w^x and y denotes the sequence of randomly initialized vectors for the words in w^y. Then the encoder of the sequence is a two-layer bidirectional long short-term memory network BiLSTM (Graves and Schmidhuber, 2005) further denoted as MT-LSTM, which produces a sequence of hidden states h:

$$h = \text{MT-LSTM}(\text{GloVe}(w^x)).$$

Next, an attention-based decoder, produces a distribution over output words $p(\hat{w}_t^y | H, w_1^y, w_2^y, \ldots, w_{t-1}^y])$ at each time-step, with H representing a stack of the elements of h along the time dimension.

The decoder is a two-layer uni-directional LSTM, initialized from the last hidden state of the encoder. For each time-step t, the decoder produces a hidden state h_t^{dec} based on the previous target embedding y_{t-1} and the context adjusted hidden state from time $t-1$, $\tilde{h}_{t-1}$:

$$h_t^{dec} = \text{LSTM}([y_{t-1}; \tilde{h}_{t-1}], h_{t-1}^{dec}).$$

Then, a vector of attention weights α_t is computed to reflect the relevance of each encoding time-step to the current decoder state in order to decide which part of the input sequence needs to be translated next:

$$\alpha_t = \text{softmax}(H(W_1 h_t^{dec} + b_1)).$$

These weights are then used to form the context-adjusted hidden state for time t, $\tilde{h}_t$:

$$\tilde{h}_t = [\text{tanh}(W_2 H^T \alpha_t + b_2); h_t^{dec}].$$

A final transformation of the context-adjusted hidden state is required to obtain the distribution over output words:

$$p(\hat{w}_t^y | H, w_1^y, w_2^y, \ldots, w_{t-1}^y]) = \text{softmax}(W_{out} \tilde{h}_t + b_{out}).$$

Once the encoder-decoder architecture is trained for the translation task, the parameters of the encoder (MT-LSTM) can be used to produce contextualized word representations for sentences involved in any new task as follows: given a sequence of words w and the sequence of the corresponding word vectors GloVe(w), the learned context vectors (CoVe) produced by the MT-LSTM are:

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe(w)}).$$

These context vectors are further used through concatenation to GloVe embeddings as input to the task at hand:

$$\tilde{w} = [\text{GloVe}(w); \text{CoVe}(w)].$$

While McCann et al. (2017) bring an important contribution by showing how transfer learning could be efficiently applied from machine translation to other NLP tasks, their experimental setup does not clearly reflect to what extent their contextualized word embeddings are informative by themselves. All reported results involve evaluations of the contextualized word embeddings concatenated with GloVe pre-trained word embeddings and/or with character-level embeddings. Moreover, both GloVe and the MT-LSTM context embeddings, were trained on large corpora (CommonCrawl-840B for GloVe and three different English-German machine translation datasets for MT-LSTM). In this respect, McCann et al. (2017) also point out that the size of the corpora used to pre-train the MT-LSTM directly correlates with the results obtained on the end tasks to which these representations are transferred. That makes their method sensitive to the size of the corpora used for pre-training which itself is also determined by the amount of available data for translation.

Following on the idea of transfer learning, recent work leverages information coming from neural language models in semi-supervised and unsupervised settings (Peters et al., 2017, 2018; Salant and Berant, 2018). Using a language modelling objective allows for unsupervised training on large amounts of data and has been shown to provide useful representations. Formally, given a sequence of n tokens $[t_1, t_2, \ldots, t_n]$, a language model computes the joint probability distribution of the sequence as follows:

$$p(t_1, t_2, \ldots, t_n) = \prod_{k=1}^{n} p(t_k | t_1, t_2, \ldots, t_{k-1}).$$

Typically, word vectors are initialized to pre-trained word embeddings or to values obtained by running variants of LSTM or CNN architectures over their characters (Ling et al., 2015; Kim et al., 2016; Labeau and Allauzen, 2017). This representation is referred to as the *context-independent* token representation and will be further denoted by x_k for the token at position k. This is then passed to a forward LSTM layer that embeds the sequence $[x_1, x_2, \ldots, x_k]$ into a fixed dimensional vector $\overrightarrow{h}_k^{LM}$. This denotes the forward language model embedding of the token at position k for the sequence $[x_1, x_2, \ldots, x_k]$ and constitutes a *context-dependent*

representation of the token. The probability of the token at position $k+1$, $p(t_{k+1})$ is then computed using a softmax layer over the words in the vocabulary:

$$p(t_{k+1}|t_1, t_2, \ldots, t_k) = \frac{\exp(z_{t_{k+1}})}{\sum_{t' \in V} \exp(z_{t'})}$$

, where V stands for the vocabulary and $z_{t_{k+1}}$ is computed as the inner product $z_{t_{k+1}} = h_k^T e_{t_{k+1}}$, with $e_{t_{k+1}}$ representing the output word embedding for the token t_{k+1} (the representation of the token in the weight matrix of the softmax layer) and h_k^T the output of the last hidden layer.

However, the above formulation only captures past context, modelling the probability of the token at position k given the history $[t_1, t_2, \ldots, t_{k-1}]$. To account for future context, one could consider a backward language model as computing the probability of the sequence

$$p(t_1, t_2, \ldots, t_n) = \prod_{k=1}^{n} p(t_k|t_{k+1}, t_{k+2}, \ldots, t_n).$$

This yields $\overleftarrow{h}_{LM}$, denoting the backward language model embedding of the token at position k for the sequence $[x_{k+1}, x_{k+2}, \ldots, x_n]$. Then the two language model embeddings for the token at postion k can be concatenated to obtain the bidirectional language model embedding for that token:

$$h_k^{LM} = [\overrightarrow{h}_k^{LM}; \overleftarrow{h}_k^{LM}]$$

Peters et al. (2017) show how to make use of the parameters learned in such a pre-trained bidirectional language model to improve the performance of a sequence tagger, by augmenting the token representations with contextual information. They consider a supervised sequence tagging model which they evaluate on the tasks of named entity recognition (NER) and chunking. Their sequence tagger (TagLM) is built on the basis of pre-trained word embeddings and enhanced with embeddings computed using the pre-trained neural language model (denoted as contextualized embeddings) as explained further: given a sequence of n tokens

$t = [t_1, t_2, \ldots, t_n]$, the representation of the k^{th} token is given by

$$x_k = [c_k; e_k]$$

$$c_k = C(t_k; \theta_c)$$

$$e_k = E(t_k; \theta_w)$$

, where c_k represents an RNN or CNN-based character representation of the word, e_k denotes the pre-trained word embedding corresponding to the word and ; is the concatenation operation. In order to obtain a contextualized representation of the input sequence, Peters et al. (2017) use $L = 2$ layers of bidirectional RNNs. Given the representation x_k for the k^{th} token as input, the bidirectional RNN at level j produces a hidden state $h_{k,j}$ such that:

$$h_{k,j} = [\overrightarrow{h}_{k,j}; \overleftarrow{h}_{k,j}]$$

$$\overrightarrow{h}_{k,j} = \overrightarrow{R}_j(x_k, \overrightarrow{h}_{k-1,j}; \theta_{\overrightarrow{R}_j})$$

$$\overleftarrow{h}_{k,j} = \overleftarrow{R}_j(x_k, \overleftarrow{h}_{k+1,j}; \theta_{\overleftarrow{R}_j})$$

, with R_j parameterized as either a GRU or an LSTM, depending on the task. Finally, the score for each tag is predicted using the output of the final layer $h_{k,L}$ followed by a dense layer. To add the contextualized representations computed using the pre-trained language model, Peters et al. (2017) propose concatenating them to the output of the first bidirectional RNN layer, namely:

$$h_{k,1} = [\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; h_k^{LM}]$$

Alternatively these contextualized representations could be added to augment the input of the first bidirectional RNN layer:

$$x_k = [c_k; e_k; h_k^{LM}]$$

or the output of the second bidirectional RNN layer:

$$h_{k,2} = [\overrightarrow{h}_{k,2}; \overleftarrow{h}_{k,2}; h_k^{LM}].$$

Experiments show however that, for the tasks considered, adding the contextualized representations to the output of the first bidirectional RNN layer achieves the highest scores. Nevertheless, other tasks may benefit more from the inclusion of these embeddings at different layers. The best results are obtained using separately trained forward and backward language models on the publicly available benchmark for large-scale language modelling, 1B Word Benchmark (Chelba et al., 2013). As previously noted for the context embeddings provided by McCann et al. (2017), it would be interesting to analyse to what extent the language modelling-based embeddings would yield the same performance by themselves in the lack of concatenation to pre-trained word embeddings or CNN-based character-level embeddings.

Peters et al. (2018) extend the idea in Peters et al. (2017) by learning a task specific linear combination of the intermediate layers of a deeper bidirectional language model pre-trained on large corpora Chelba et al. (2013). They denote their embeddings ELMo (Embeddings from Language Models) and evaluate them on a variety of tasks ranging from sentiment analysis, textual entailment recognition and question answering to co-reference resolution, semantic role labeling and named entity extraction. Their approach follows closely that of Peters et al. (2017), with some extensions as further detailed.

As previously, given a sequence of n tokens $t = [t_1, t_2, \ldots, t_n]$, x_k denotes the context-independent representation of the token t_k obtained either through initialization to its corresponding pre-trained word embedding or via a CNN over its characters. This representation is then fed into an L-layer bidirectional language model that outputs a context-dependent representation $h_{k,L}^{LM}$ for a given token t_k. For each layer j with $j = 1, \ldots, L$, $\overrightarrow{h}_{k,j}^{LM}$ represents the forward language model embeddings of t_k obtained from layer j, and $\overleftarrow{h}_{k,j}^{LM}$ is the backward language model embeddings of t_k obtained from layer j. Thus, the bidirectional language model embedding

obtained from layer j is given by

$$h_{k,j}^{LM} = [\overrightarrow{h}_{k,j}^{LM}, \overleftarrow{h}_{k,j}^{LM}], \forall j = 1, \ldots, L.$$

Differently than in the work of Peters et al. (2017), some of the parameters for the forward and backward bidirectional language models are shared, namely the parameters of the token representations θ_x and the softmax layer parameters θ_s. Thus, the bidirectional language model jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{n} (\log p(t_k | t_1, \ldots, t_{k-1}; \theta_x; \overrightarrow{\theta}_{LSTM}; \theta_s) + \log p(t_k | t_{k+1}, \ldots, t_n; \theta_x; \overleftarrow{\theta}_{LSTM}; \theta_s)).$$

Further, the ELMo embeddings are computed as a task specific combination of the intermediate layers in the bidirectional language model. More precisely, given a token t_k, the L-layer bidirectional language model computes a set of representations:

$$\begin{aligned} R_k &= \{x_k^{LM}, h_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{h_{k,j}^{LM} \mid j = 0, \ldots, L\} \end{aligned}$$

Then, the ELMo representation is given by a task specific combination of all bidirectional language model layers:

$$\text{ELMo}_k^{task} = E(R_k; \theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}.$$

This formulation enables Peters et al. (2018) to analyze the contribution of different layers of the deep bidirectional language models to each considered task. However, it also implies that their representation of tokens is task dependent: although the intermediate layers representations are task independent, their linear combination is learned with respect to the task at hand. Addi-

tionally, the authors note that in most cases high performance is ensured by fine-tuning the parameters of the pre-trained bidirectional language model on the training data, before fixing them during task training. Similarly to the protocol in McCann et al. (2017), the token representations obtained from the bidirectional language model are concatenated with the context-independent token representation x_k, (obtained from pre-trained word embeddings and/or character-based representations[1]) before being fed as input to any given task:

$$[x_k; \text{ELMo}_k^{task}].$$

Alternatively, results are improved for some tasks when adding the token representations to the output of the task RNN as:

$$[h_k; \text{ELMo}_k^{task}].$$

Continuing on the same line, Salant and Berant (2018) show that adding rich contextualized representations derived from language modelling to a basic model for question-answering achieves state-of-the-art results. Moreover, they show that the model covering only minimal question-document interaction, yet making use of contextualized token representations, is able to surpass more sophisticated models that focus on the interaction between the question and the document. They propose a *re-embedding* approach in which the model is allowed to choose between the context-dependent and the context-independent representations of a token, through a gating mechanism.

Given a sequence of n tokens $[t_1, t_2, \ldots, t_n]$, let e_k denote the pre-trained word embeddings corresponding to token t_k and c_k denote the character-based representation of the same token obtained using a CNN over character embeddings (Kim et al., 2016).

Then the re-embeddings of token t_k is the result of a Highway layer (Srivastava et al., 2015)

[1]The experiments considered in Peters et al. (2018) use pre-trained GloVe embeddings (Pennington et al., 2014) concatenated to character-level embeddings: 2048 character n-gram convolutional filters, followed by two highway layers (Srivastava et al., 2015) and a projection to dimension 512.

formulated as follows:

$$t'_k = g_k \odot e_k + (1 - g_k) \odot z_k$$

$$g_k = \sigma(W_g x_k + U_g u_k)$$

$$z_k = tanh(W_z x_k + U_z u_k)$$

$$x_k = [e_k; c_k]$$

, where $\odot$ represents the element-wise product operator, W_g,W_z,U_g and U_z are parameter matrices, x_k denotes the non-contextualized representation of the token and u_k stands for a contextualized representation of t_k.

Salant and Berant (2018) provide two ways of constructing the contextualized representation u_k for any given token t_k: either using the hidden states of the top layer in a stacked bidirectional LSTM

$$\begin{aligned}\{u_1, u_2, \ldots, u_n\} &= \textbf{BiLSTM}(x_1, x_2, \ldots, x_n) \\ &= \{h_1, h_2, \ldots, h_n\}\end{aligned}$$

or by leveraging information coming from language modelling and concatenating such a representation to the hidden states mentioned previously

$$u_k = [h_k; o_k].$$

o_k thus represents the embedding of token t_k as provided by the hidden states of the trained language model. Similarly to one of the settings presented in Peters et al. (2017), they use the language model of Józefowicz et al. (2016), pre-trained on the 1B Word Benchmark corpus (Chelba et al., 2013). This language model consists of an initial layer of character-based word representations, two stacked forward LSTM layers and a softmax prediction layer.

In their work, Salant and Berant (2018) also emphasize the need of complementing the infor-

mation coming from the embedding of rare words with contextualized representations. They observe that the activations of the gate g_k have smaller values for rare words. In other words, the re-embedded representation of a token t'_k (corresponding to a rare word) is constructed using to a lesser degree its pre-trained word embedding and relies mostly on its contextualized representation. Overall, their findings are interesting as they validate further the importance of having contextually-informed features as input to deep learning models, even when these models have simple architectures.

Recently, Devlin et al. (2019) have also provided a method (denoted BERT) to obtain token embeddings by leveraging an architecture based on multi-layer bidirectional Transformers (Vaswani et al., 2017).

As previously noted, most of these token embeddings are constructed either in a supervised or semi-supervised manner using large corpora and a large parameter space, which contrasts to the current proposal. Moreover, within these models, it has been shown that the amount of data used for constructing representations correlates with increases in performance, making training on large amounts of data a pre-requisite for the success of such approaches: CoVe (McCann et al., 2017) uses 350M words corpus for its best performing model, ELMo (Peters et al., 2018) uses a 1B words corpus, while BERT (Devlin et al., 2019) leverages a 3.3B words corpus and an architecture of 340M parameters. In contrast to these methods, the proposal in the current thesis can provide competitive results even when leveraging only the corpus at hand to construct representations for its tokens.

Furthermore, none of the related work provides an explicit encoding of the sentence structure into the token embeddings themselves by directly leveraging information from syntactic dependencies or a flexible framework for explicitly incorporating linguistic knowledge. Additionally, in most of the evaluations of the related work, the proposed token embeddings are concatenated to pre-trained word embeddings and/or other feature vectors before being fed as input for an end task. This last aspect makes it difficult to quantify the contribution of the token embeddings themselves to the obtained results.

3.2 SATokE – Syntax-Aware Token Embeddings

Following the success of tensor factorisation models at learning latent representations of entities and relations from a graph, we propose a model that makes use of tensor factorisation to produce token embeddings for sentences and their syntactic structures. The proposed token embeddings have the benefit of encoding the information about the structure of the sentence from a dependency point of view in the representations themselves and not in the parameters of an end task model they may be fed as input to. As a byproduct, our method also provides embeddings for the relations present between the tokens in a parse tree, which can be further leveraged to compute token embeddings on new corpora. Furthermore the token and relations embeddings are constructed in an unsupervised manner and can be obtained for any given corpus regardless of its size, provided access to the parse trees of the sentences in that corpus.

The token embeddings we propose will be compared empirically to context-independent word type embeddings pre-trained on large corpora on local contexts (Mikolov et al., 2013b; Pennington et al., 2014) in Section 3.4 on a series of tasks ranging from sentiment analysis to textual entailment recognition and implicit discourse relation classification (Popa et al., 2019a,b). We additionally provide a comparison to word type embeddings pre-trained on syntactic information (Levy and Goldberg, 2014) and to an alternative token embeddings method proposed in the literature (Peters et al., 2018).

3.2.1 Preliminaries

We propose to learn token embeddings in an unsupervised manner by choosing to model sentences as graphs: the nodes in each graph represent the individual tokens in the sentence and graph edges stand for the adjacency and syntactic relations between these tokens, as given by a parse tree of the sentence.

To learn these token embeddings, we make use of tensor factorisation, through a model that can

be seen at the intersection between the RESCAL model (Nickel et al., 2011) and the TransE model (Bordes et al., 2013), both of which have been proposed for learning embeddings in large relational databases. The structure used to model relations between entities in the current work follows that proposed in RESCAL, with entities represented by vectors and relations by matrices. Similarly to TransE, we choose to optimise a ranking loss function, unlike RESCAL that proposes a squared loss optimisation in its original version or other related literature that uses logistic loss (Trouillon et al., 2016).

Altough RESCAL is designed as a model of large relational databases, with large number of entities and relations between them, our case is different. The entities are partitioned into sentences and no relations can exist between entities belonging to different sentences. This makes the learning more challenging. Unlike other multi-relational approaches where a threshold is typically set on the number of times entities appear in relations in order to ease the learning (Bordes et al., 2013), this is not a feasible option in our case. The maximum number of times an entity takes part in a relation is limited to a very low value which is dependent on the parse tree that entity is part of. The goal of learning in our model is to abstract away from the individual sentences and learn the underlying regularity of parse trees. Thus, unlike in a typical relational learning scenario, the proposed model learns from many small graphs rather than from one single big one.

The factorisation model works by optimising a reconstruction of the graphs' tensors, so that given the token embeddings one can reconstruct each labelled edge in the graph. To predict the different relations, it learns one real-valued matrix per relation label. It is these relation matrices which capture the regularities which generalise across sentences. The information about the individual sentence is captured in the token embedding vectors. Thus at the end of the decomposition we aim to obtain a tensor of relation embeddings with one real-valued matrix per relation and a matrix of token embeddings with one real-valued vector per token.

Because each token, in conjunction with the relation matrices, needs to be able to reconstruct its relations with other tokens, the token vectors encode information about the graph context

in which they appear. In this way, each token representation can be interpreted as a compressed view of the sentence's entire graph from the point of view of that token. The final sentence representation is a set of all individual token representations which, by construction, are contextually-aware and syntactically-aware.

Additionally to the tensor decomposition, our model incorporates another part of the loss that constrains the nature of the entity representations that we wish to obtain. In order to account for the semantic aspect, to enable information sharing across sentences and to make use of the distributional information inferred from a large corpus, we constrain the token embeddings to be similar to a pre-trained representation of the word types they denote. Although we could also be learning the word type embeddings from scratch, one advantage of such an approach is the possibility to learn tokens embeddings for any dataset regardless of its size.

3.2.2 Unsupervised learning of token embeddings via tensor factorisation

In the following we present our proposed method for token embeddings computation. This method can be applied to any dataset of sentences with the sole prerequisites of having access to parsing information and to pre-trained general purpose word type embeddings. Alternatively, word type embeddings can also be computed using this method, provided a sufficiently large amount of data is available.

There are two possibilities for token embedding computation: either a 1-step setting or a 2-steps setting. In the *1-step setting*, given a new dataset of sentences and their parses, we learn a representation for each word token in the sentences as well as for each relation holding between these tokens from scratch - in the current work we consider syntactic relations from the parse tree as well as adjacency relations. We expect the token embeddings obtained through such a learning scenario to perform best as they are directly learned for the given dataset along with the relations present in the dataset.

An alternative to this is the *2-step setting* in which we first learn the relation embeddings on one

corpus and then we fix them when computing token embeddings on another corpus. This approach reduces the computation time as it leverages relation embeddings previously learned on a different corpus. However, it may also affect the quality of the induced token representations as the relations are kept fixed to values previously learned and there may be a domain mismatch between the two corpora on which the relations and the tokens were learned. Given a new dataset of sentences, their parses and previously trained relations embeddings, one can learn token embeddings for the dataset through the same process by fixing the relation embeddings to the pre-trained ones and optimising only for the token embeddings.

For completeness, we will further present the method corresponding to the 1-step setting, but we will discuss results corresponding to the 2-step setting as well. We hypothesize (and later show empirically) that the token embeddings learned in the 1-step setting obtain better results than those learned in an 2-step setting, yet token embeddings learned in both settings outperform general purpose word type embeddings on the sentence understanding tasks considered.

Formally, given the graph $G^{(s)}$ of a sentence s as provided by its parse tree, we model the interactions between the tokens in the sentence as a 3-dimensional tensor $T^{(s)}$. In this work, the possible relations between tokens are given by the parse tree of the sentence along with an additional adjacency relation to denote neighbouring tokens. However, the current proposal can be extended to deal with additional relations coming from semantic parsers for example. All relations are modelled using asymmetric matrices. Each matrix in the tensor corresponds, thus, to one of the possible relations between the tokens: this enables modelling the different interaction types between tokens using different matrices. Let $tok_i^{(s)}$ represent the token that appears at position i in the sentence s and let $M_{REL}^{(s)}$ denote the matrix in $T^{(s)}$ that holds entries related to the REL relation in sentence s. Then an entry t_{ijk} in the final tensor $T^{(s)}$ takes value 1 if the relation k holds between the token $tok_i^{(s)}$ and the token $tok_j^{(s)}$ such that the token $tok_i^{(s)}$ is the head in the dependency relation and $tok_j^{(s)}$ is the dependant. The matrix corresponding to the adjacency information is populated similarly with value 1 whenever the token at position i precedes the token at position j in the given sequence. All the remaining entries in the tensor

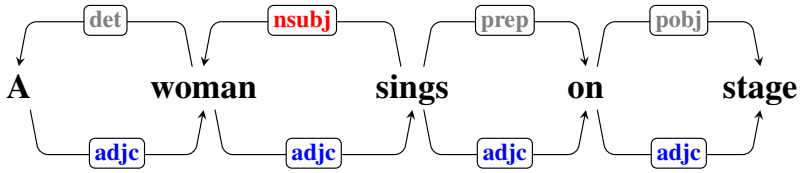


Figure 3.1: Example of relations that hold for one sentence: relations from the dependency parse (in grey) + adjacency relations (in blue).

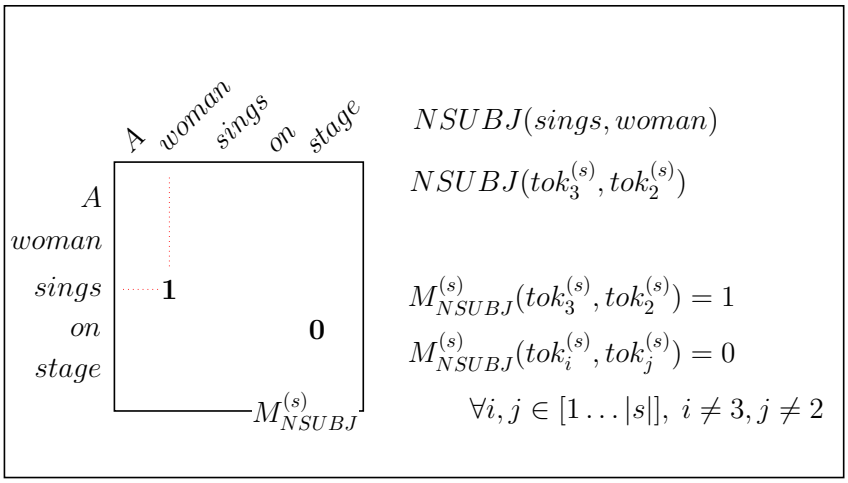


Figure 3.2: Example of a matrix corresponding to the dependency relation NSUBJ for sentence s.

$T^{(s)}$ are set to value 0.

$$t_{ijk} = 1 \Leftrightarrow M_k^{(s)}(i,j) = 1$$

$$t_{ijk} = 0 \Leftrightarrow M_k^{(s)}(i,j) = 0$$

An example for a given sentence s is presented in Figure 3.1. This includes the sentence parse information (marked in gray and red) as well as the adjacency information (marked in blue). In the given example, the $NSUBJ$ relation holds between $tok_3^{(s)}$ and $tok_2^{(s)}$, with $tok_3^{(s)}$ being the head of the relation. Thus the matrix corresponding to the $NSUBJ$ relation in sentence s will have entry 1 at position $M^{(s)}_{SUBJ}(tok_3^{(s)}, tok_2^{(s)})$ and 0 everywhere else, as shown in Figure 3.2. Similarly, all the other matrices corresponding to syntactic dependency relations are populated based on the information from the sentence dependency parse tree.

For encoding information about neighbouring tokens in sentence s, an assymmetric matrix $M^{(s)}_{ADJC}$ is used as shown in Figure 3.3. This matrix holds value 1 at position $M^{(s)}_{ADJC}(tok_i^{(s)}, tok_{i+1}^{(s)})$

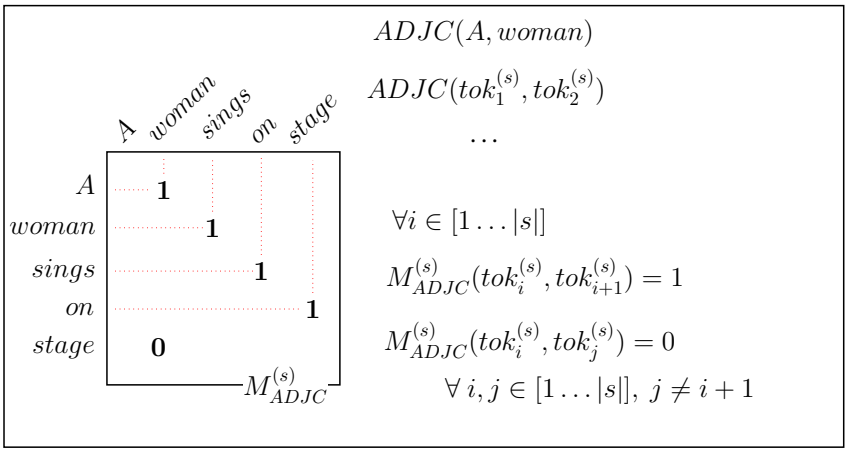


Figure 3.3: Example of a matrix corresponding to the adjacency relation ADJC.

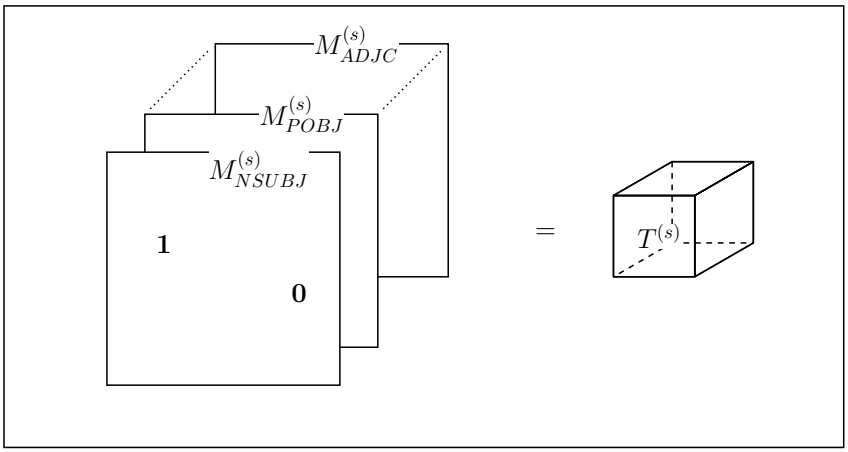


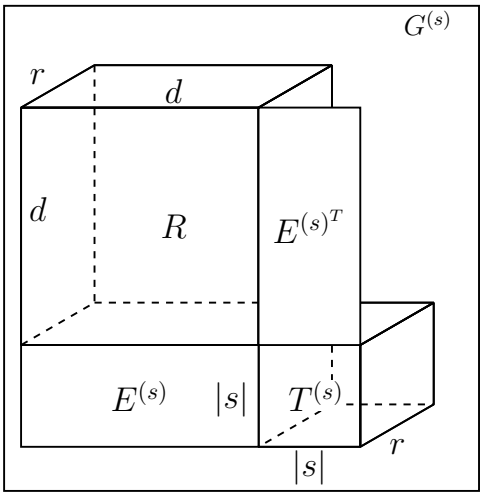
Figure 3.4: Matrices form the tensor $T^{(s)}$ for sentence s.

and 0 everywhere else, $\forall\, i \in [1 \dots |s|]$.

For sentence s, the matrices corresponding to syntactic dependency relations along with the adjacency matrix form the tensor $T^{(s)}$ as shown in Firgure 3.4. Such a tensor is formed for each of the sentences in a given dataset.

Further, in order to obtain embeddings for all the tokens in a sentence as well as for all the relations holding between these tokens, we factorise each $T^{(s)}$ into:

- $E^{(s)}$, a matrix holding all token embeddings for sentence s, with one token embedding per row
- R, a tensor for all the relations (shared between all sentences), with one matrix per relation



(a)

$G^{(s)}$	graph of sentence s
$T^{(s)} : r \times \|s\| \times \|s\|$	relations between tokens in $G^{(s)}$
$R : r \times d \times d$	relation tensor embeddings
$E^{(s)} : \|s\| \times d$	vector embeddings for tokens in $G^{(s)}$
d	size of the embedding vectors
r	number of binary relations
$\|s\|$	number of tokens in sentence s

$$\forall s,\ T^{(s)} \approx E^{(s)} \cdot R \cdot E^{(s)^T}$$

(b)

Figure 3.5: Sentence graph decomposition

embedding.

The factorisation has to be such that we can recompute the various relations between the tokens in each sentence:

$$\forall s,\ T^{(s)} \approx E^{(s)} \cdot R \cdot E^{(s)^T}$$

The decomposition is shown in Figure 3.5.

However, constraining an exact reconstruction of a binary tensor (with 1s for existing relations between tokens and 0s otherwise) may be too strict. Therefore, as objective function, we optimise a ranking loss within the tensor ($T^{(s)}_{loss}$) that aims at scoring the reconstruction of positive triples $t^{(s)}_{ijk} = (e^{(s)}_i, R_k, e^{(s)}_j)$ higher than that of negative ones. We thus relax the exact recon-

struction constraint to a ranking one.

We use an additional regularisation term ($R^{(s)}_{loss}$) to minimise the gap between the token embeddings representation and the word type representation of their words. Conceptually, this is similar to the vector space preservation term in Mrkšic et al. (2016) that controls how much the token embeddings can deviate from their corresponding word representations. The overall goal is to create embeddings that are close, through $R^{(s)}_{loss}$, to the original word embeddings (known to capture semantics) and at the same time are syntactically-informed, through $T^{(s)}_{loss}$, so as to capture fine-grained semantic differences according to the role a given word plays in a sentence. Optimising only $T^{(s)}_{loss}$ would be insufficient as it would lack the notion of semantics provided through $R^{(s)}_{loss}$. Indeed, by imposing that the token embeddings be close to the word embeddings, one also forces the semantic representation of the word embeddings to be shared across all sentences.

The optimisation problem is formulated as

$$\min \sum_{s \in S} \alpha_R(T^{(s)}_{loss}) + (1 - \alpha_R)(R^{(s)}_{loss}) \tag{3.1}$$

where:

$$T^{(s)}_{loss} = \sum_{\substack{t^{(s)}_{ijk} \in G^{(s)}, \\ t^{(s)}_{i'j'k'} \in \neg(t^{(s)}_{ijk})}} \max(0, \gamma + \langle e^{(s)}_{i'}, R_{k'}, e^{(s)}_{j'} \rangle - \langle e^{(s)}_{i}, R_{k}, e^{(s)}_{j} \rangle)$$

and

$$R^{(s)}_{loss} = \sum_{e^{(s)}_i \in G^{(s)}} -\log \sigma(e^{(s)}_i \cdot w^{(s)}_i)$$

with $G^{(s)}$ the graph of sentence s holding all tokens and all relations present in the sentence, $e^{(s)}_i$ the token embedding of token i in sentence s, R_k the matrix embedding for the relation k, $w^{(s)}_i$ the pre-trained word embedding corresponding to the token $e^{(s)}_i$, γ the margin hyperparameter and $\neg(t^{(s)}_{ijk})$ the set of negative triples associated with t_{ijk} as explained below.

d denotes the dimensionality of the embeddings, while r denotes the number of relations. Thus

the bilinear product:

$$\langle e_i^{(s)}, R_k, e_j^{(s)} \rangle = e_i^{(s)\ 1\times d} \cdot R_k^{d\times d} \cdot e_j^{(s)^T\ d\times 1}.$$

Considering all tokens in one sentence s, that results in approximating the relations tensor for sentence s, $T^{(s)}$ by

$$E^{(s)\ (|s|\times d)} \cdot R^{(d\times r\times d)} \cdot E^{(s)^T\ (d\times |s|)}.$$

The regularization term $R_{loss}^{(s)}$ can be seen as a particular case of cross entropy[2]

$$R_{loss}^{(s)} = -y \cdot \log \hat{y} - (1-y) \cdot \log(1-\hat{y})$$

with $y = 1$ and $\hat{y} = \sigma(e_i^{(s)} \cdot w_i^{(s)})$, where σ denotes the sigmoid function.

As detailed in the corresponding implementation details sections for each task, we explore multiple settings by varying the ratio of negative sampling for one positive triple. A negatively sampled example in the tensor is obtained by altering one element of the triple while fixing the remaining two: this element can be either one of the entities or the relation holding between them. As mentioned above, given a triple $t_{ijk}^{(s)} = (e_i^{(s)}, R_k, e_j^{(s)})$, we denote by $\neg(t_{ijk}^{(s)})$ the set of negative examples associated to it. We consider here that $\neg(t_{ijk}^{(s)})$ is formed of the following elements:

$$\neg(t_{ijk}^{(s)}) = \{(e_{i'}^{(s)}, R_k, e_j^{(s)}), (e_i^{(s)}, R_{k'}, e_j^{(s)}), (e_i^{(s)}, R_k, e_{j'}^{(s)})\} \forall i' \neq i, j' \neq j, k' \neq k.$$

Ideally we would like to sample not only the positive triples, but also all the negative ones. However, this is computationally expensive, which is why we choose to perform this contrastive sampling which is more informative than a random sampling strategy.

We optimise Eq.(3.1) using mini-batch stochastic gradient descent. We construct our batches b_m such that if $t_{ijk}^{(s)} \in b_m$ then $\neg(t_{ijk}^{(s)}) \subset b_m$. Sampling positive points and their negative

[2]An alternative was to use L2 between the token embeddings $e_i^{(s)}$ and their corresponding word type embeddings $w_i^{(s)}$, but preliminary experiments resulted in decreased performance.

counterparts together is important to ensure the consistency of the information at the basis of which a step is taken when optimising each mini-batch. Further details regarding how the validation set is constructed are provided with the implementation details in Section 3.3.2.

3.3 Experimental protocol

3.3.1 End task architectures

The set of token embeddings $e_i^{(s)}$ which result from the factorisation presented in 3.2 constitutes the token-based representation of a sentence. Alternatively, a sentence can be represented as a set of pre-trained word type embeddings, such as GloVe (Pennington et al., 2014), Word2Vec (Mikolov et al., 2013b) or dependency-based word embeddings (Levy and Goldberg, 2014).

Such a representation is further fed as input to a neural network architecture. In this work we have explored the two approaches that correspond to the building blocks of models mainly employed in the literature. This first approach is using a recurrent neural network architecture with Long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) to encode a sentence and the second approach uses a convolutional neural network (CNN) encoder following Kim (2014).

Given an input sequence of k tokens

$$t = [t_1, t_2, \ldots, t_k]$$

let $\boldsymbol{x}_i$ denote the vector representation that corresponds to token t_i. This vector can be either the corresponding pre-trained word embedding representation of the token or it can be the computed representation using the method in 3.2. Then the vectorial representation of the sequence is given by

$$[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k].$$

Long short-term memory

One of the most widely used models to encode sequences, that addresses the issue of learning long-term dependencies and that takes into account contextual information, is the long-short term memory network, a variant of the recurrent neural network. We adopt such an architecture to model each sentence. Given the input sequence of vectors $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$, an LSTM computes the hidden state sequence representation

$$[\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_k].$$

At each time step i, the model reads $\boldsymbol{x}_i$ as input and updates the h_i hidden state as follows:

$$\boldsymbol{i}_i = \sigma(\boldsymbol{W}^{xi}\boldsymbol{x}_i + \boldsymbol{W}^{hi}\boldsymbol{h}_{i-1} + \boldsymbol{b}^i)$$

$$\boldsymbol{f}_i = \sigma(\boldsymbol{W}^{xf}\boldsymbol{x}_i + \boldsymbol{W}^{hf}\boldsymbol{h}_{i-1} + \boldsymbol{b}^f)$$

$$\boldsymbol{o}_i = \sigma(\boldsymbol{W}^{xo}\boldsymbol{x}_i + \boldsymbol{W}^{ho}\boldsymbol{h}_{i-1} + \boldsymbol{b}^o)$$

$$\boldsymbol{c}_i = \boldsymbol{f}_i \odot \boldsymbol{c}_{i-1} + \boldsymbol{i}_i \odot \tanh(\boldsymbol{W}^{xc}\boldsymbol{x}_i + \boldsymbol{W}^{hc}\boldsymbol{h}_{i-1} + \boldsymbol{b}^c)$$

$$\boldsymbol{h}_i = \boldsymbol{o}_i \odot \tanh \boldsymbol{c_i}$$

with $\odot$ representing the element-wise product and σ the sigmoid function. $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$ are the input gate, forget gate and output gate respectively and $\boldsymbol{c}$ is the memory cell. $\boldsymbol{W}^{xi}$, $\boldsymbol{W}^{xf}$, $\boldsymbol{W}^{xo}$, $\boldsymbol{W}^{hi}$, $\boldsymbol{W}^{hf}$, $\boldsymbol{W}^{ho}$, $\boldsymbol{W}^{xc}$, $\boldsymbol{W}^{hc}$, $\boldsymbol{b}^i$, $\boldsymbol{b}^f$, $\boldsymbol{b}^o$, $\boldsymbol{b}^c$ are parameters of the model. The final representations for the sequence is then given by the last hidden state representation $\boldsymbol{h}_k$.

Convolutional neural networks

In order to represent a sentence using convolutional neural networks (CNNs), we follow the approach of Kim (2014). The general architecture is shown in Figure 3.6. Given the input se-

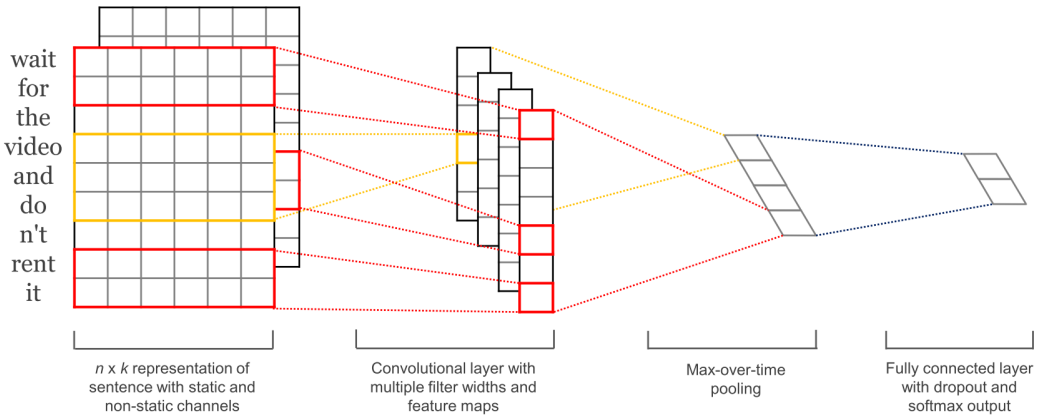


Figure 3.6: General CNN architecture (Kim, 2014)

quence of vectors $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k]$, let $\boldsymbol{x_{i:i+j}}$ be the concatenation of word vectors $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+j}$ and $\boldsymbol{W} \in \mathbb{R}^{h \times d}$ be a filter applied to a window of h words to produce a feature c_i. Let b be a bias term and f a non-linear function. Then

$$c_i = f(\boldsymbol{W} \cdot \boldsymbol{x}_{i:i+h-1} + b).$$

A feature map $\mathbf{c} \in \mathbb{R}^{n-h+1}$ is created by applying the filter to each possible window of words in the argument $\{\boldsymbol{x}_{1:h}, \boldsymbol{x}_{2:h+1}, \ldots, \boldsymbol{x}_{k-h+1:k}\}$. Thus

$$\mathbf{c} = [c_1, c_2, ..., c_{k-h+1}].$$

This is followed by a max-pooling operation $\hat{c} = max(\mathbf{c})$ to obtain the most important feature, i.e. the one with the highest value, corresponding to the particular filter. For m filters with different window sizes we obtain m features: $z = [\hat{c}_1, ..., \hat{c}_m]$. The representation of the input sequence is thus the m-dimensional vector $\boldsymbol{z}$.

Enhancing representations with positional information

One way to inject into word representations information about their order in a sequence is through **positional encodings** (Gehring et al., 2017). Following the approach of Vaswani et al.

(2017), we use fixed sinusoidal positional encodings of the same dimension as the word embeddings and sum them together. Given an input sequence of word embeddings

$$[x_1, x_2, ..., x_k]$$

with $x_j \in \mathbb{R}^{\mathrm{d}}$ and the positional information for each of the words

$$P = (p_1, p_2, ..., p_k)$$

with $p_j \in \mathbb{R}^{\mathrm{d}}$, our new sequence representation will be

$$[x_1 + p_1, x_2 + p_2, ..., x_k + p_k].$$

For the positional encodings we use the sine and cosine functions:

$$p(\text{pos}, 2i) = \sin(\text{pos}/10000^{2i/d})$$

$$p(\text{pos}, 2i + 1) = \cos(\text{pos}/10000^{2i/d})$$

with *pos* representing the position, *i* the dimension and *d* the embedding size.

Self-attention has also been recently used in a variety of tasks. It has been shown to be a useful mechanism to connect information from different positions of a sequence in order to better represent it. Similarly to Vaswani et al. (2017) we use a dot-product attention:

$$Att(Q, K, V) = softmax(QK^T)V$$

where Q, K and V are transformations of the input embeddings

$$[x_1, x_2, ..., x_k]$$

through matrices W_Q, W_K and W_V. Each of the obtained vectors computed through the attention mechanism represents thus a weighted combination of all tokens' representations in the sequence. Further, these representations constitute the input to the LSTM and CNN architectures.

3.3.2 Implementation details

Pre-processing and linguistic information

The sentences in all considered datasets are parsed and further encoded using the method described in 3.2. The dependency information used in the current work is obtained with the spaCy toolkit [3], leveraging the transition-based dependency parser of Honnibal and Johnson (2015) and using the CLEAR label scheme for the syntactic dependencies. For each dataset two different learning strategies are explored: either learning token embeddings and relation embeddings from scratch (1-step approach) or pre-training the relation embeddings on a corpus and then fixing them to infer token embeddings for all the datasets considered (2-steps approach).

Token embeddings are computed for all the sentences in the datasets used. In this process a threshold is set on the frequency of words th_W to take into account (here set to 2). All words that appear in the corpus with a frequency lower than the threshold or that do not have a corresponding pre-trained word type embedding (here the GloVe (Pennington et al., 2014) word type embeddings are used) are set to an 'unknown_pos' tag, where pos stands for the part of speech associated to the word. The representations of these word types are also learned along with the token representations. Embeddings for punctuation signs are also used when provided by GloVe and replaced by a general 'unknown_puncttag' term otherwise, which is further trained along with the graph of the sentence.

An additional threshold is imposed on the frequency of syntactic dependencies th_D to take into account (here set to 1000). Infrequent relations are replaced by an 'unknown_relation' tag and

[3] https://spacy.io/

learned along with the rest of the graph. Apart from the syntactic dependencies obtained from the parse tree, one additional relation is added to the parse graph in order to encode adjacency information. The frequency distributions of all syntactic relations considered for each dataset (after applying the threshold th_D) are presented in Figures 3.8 to 3.17.

SATokE Token embeddings computation

For each dataset multiple threads of token embeddings computation are run considering varying initial learning rate values in the range [10^{-2},10^{-5}] and negative sampling factors (1x, 5x, 10x). In all cases the ratio between the two losses α_R is set to 0.5 and mini-batches are fixed to size 300. Considering all dependency relations with a frequency higher than th_D in the corpus corresponds to using the top 12 to 35 most frequent relations, depending on the corpus.

For assessing the embeddings and to aid in the optimisation process, the mean reciprocal rank (MRR) is used, which is a standard metric for the evaluation of the quality of triple ranking (Bordes et al., 2013).

At the beginning all token embeddings are randomly initialised [4]. Then the optimisation is performed using Adam (Kingma and Ba, 2014). During this process, negative triples are re-sampled for each considered positive triple at every epoch. The criteria for ending the optimisation is based on early stopping, considering the MRR score on the validation set.

To construct the validation set 20% of the positive triples are randomly sampled from the training set $t^{(s)}_{ijk} = (e^{(s)}_i, R_k, e^{(s)}_j) \in T_+$ such that at least one of the two following conditions holds:

$$\exists R_{k''}, e^{(s)}_{j''} \in G^{(s)} : (e^{(s)}_i, R_{k''}, e^{(s)}_{j''}) \in T_+$$

or

$$\exists e^{(s)}_{i''}, R_{k''} \in G^{(s)} : (e^{(s)}_{i''}, R_{k''}, e^{(s)}_j) \in T_+$$

[4]Preliminary experiments with token embeddings initialised with their corresponding GloVe word type embeddings did not result in improvements.

where $G^{(s)}$ denotes the graph of the sentence composed of all the entities and relations present in the sentence and T_+ is the set of all positive triples in the dataset. This prevents the separation of the graph of a sentence into two disjoint parts for training and validation. This allows us to learn embeddings for all entities as it enables information from the entities in the training set to flow to the entities in the validation set via the relations.

1-step vs 2-steps learning setups

As mentioned in 3.2, one can leverage pre-trained relations embeddings in order to compute token embeddings for any new dataset. In order to asses the quality of token embeddings induced using pre-trained relations embeddings, a 2-steps process is employed: the model is first used to train relations embeddings on the Penn TreeBank corpus of about 40k sentences (Marcus et al., 1993) and then these embeddings are fixed in order to learn token embeddings for each dataset. In practice, given any new sentence, one can use the 2-step setup to compute its token representations based on pre-trained relations embeddings.

For training the relation embeddings, the margin parameter is set to $\gamma = 1$, the initial learning rate value is varied in the range [10^{-3},10^{-4}], the ratio between the two losses is set to $\alpha = 0.5$, mini-batches are of size 300 and the negative sampling factors remain (1x, 5x, 10x). Then token embeddings are computed for each dataset using the same hyperparameters as the ones used for learning the relations they are inferred from. The results obtained using token embeddings computed in the 2-step setup are presented and discussed in Section 3.4.5.

End task learning

The details on the hyper-parameters used for both the LSTM and CNN encoders are reported in the corresponding implementation details subsections for each considered task. With both encoding schemes, the sequence representation is further passed to a fully connected layer that outputs the probability distribution over labels: $y = W \cdot q + b$, where q stands for the represen-

tation h_k in the case of an LSTM encoder or z in the case of a CNN encoder (for single sentence classification tasks) or the concatenation of these representations (for sentence pair classification tasks). The training objective is defined as the cross-entropy loss between the output of the softmax layer and the class labels.

The general architecture for sentence pair classification can be seen in Figure 3.7. The architecture for single sentence classification follows the same structure as the one in Figure 3.7, but for one single sentence as input. Sentences are encoded using the methods described in Section 3.3.1 (Encoded S_1, Encoded S_2). For the sentence pair classification, the representations of the two sentences are concatenated to be provided as input to a fully connected layer before the softmax layer.

This architecture enables the exploration of the behaviour of the produced token embeddings on a large and representative set of tasks: textual entailment prediction, paraphrase detection, sentence polarity detection, subjectivity detection and question-type categorisation.

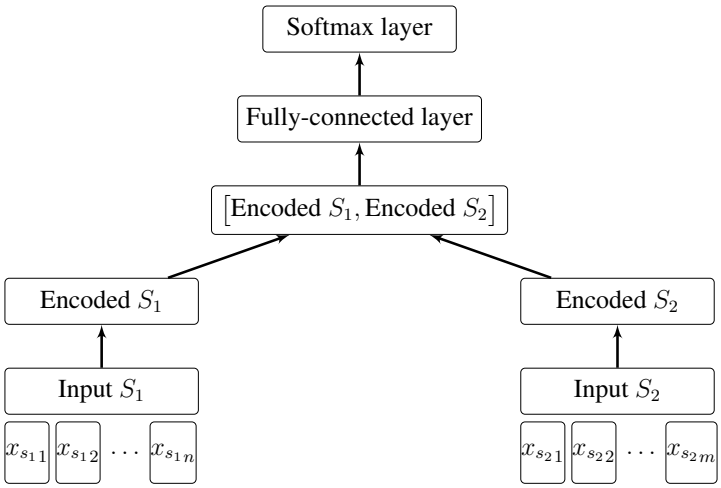


Figure 3.7: General architecture for sentence pair classification

Throughout all the experiments the input embeddings are fixed to be one of the following: the popular pre-trained word type embeddings GloVe (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013b), the dependency-informed word type embeddings Dep-based WE (Levy and Goldberg, 2014), the ELMo token embeddings (Peters et al., 2018) or the proposed computed token embeddings, SATokE.

The pre-trained word type embeddings of Pennington et al. (2014) considered in the current work are 300-dimensional and were learned from the Common Crawl corpus[5]. These are used for constraining the semantics of the SATokE token embeddings through $R^{(s)}_{loss}$ as discussed in Section 3.2. They are also used for evaluation on each dataset in Section 3.4. The Word2Vec word type embeddings used are 300-dimensional and were trained on part of the Google News dataset (about 100 billion words) [6] using the Skip-gram model (Mikolov et al., 2013a,b). The pre-trained word type embeddings of Levy and Goldberg (2014) are also 300-dimensional and are learned using the Skip-Gram model (Mikolov et al., 2013a,b), from English Wikipedia, considering dependency-based contexts[7].

Results are provided using two variants of the ELMo token embeddings (Peters et al., 2018): the 1024-dimensional embeddings that obtained the best results in Peters et al. (2018) as well as the 256-dimensional embeddings obtained from the pre-trained ELMo models[8] (for fair comparison to the proposed 300-dimensional SATokE). The pre-trained ELMo models are used as they represent the optimized versions shown to provide the best results (Peters et al., 2018). The ELMo models considered were trained on the 1 Billion Word Benchmark consisting of approximately 800M tokens of news crawl data from WMT 2011.

Fixing the embeddings prevents their adaptation to the end task and disallows the representations of frequent words in the training data to distance themselves in the embedding space from representations of related but rare words. It also enables to perform a stricter assessment of the difference between the word type embeddings and the token embeddings as input features. For the embeddings that lead to the best results, an additional set of experiments considers allowing the input embeddings to change. Such experiments will be denoted by the usage of the suffix *(fine-tuned)* after the embeddings type. More particularly, in the case of SATokE embeddings, the suffix *(fine-tuned)* indicates that the construction of the embeddings was conditioned on GloVe word type embeddings that were previously fine-tuned for the end tasks.

[5]https://nlp.stanford.edu/projects/glove/
[6]https://code.google.com/archive/p/word2vec/
[7]https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/
[8]https://allennlp.org/elmo

To tune all hyper-parameters for each end task learning scenario, a development set is used when available or 20% of the training set is randomly selected when such a development set is not provided. The optimisation stops when the loss increases on the development set (with a patience of 5). Adam (Kingma and Ba, 2014) was used for the optimisation and the hyper-parameters that lead to the best results for each task are reported in their corresponding sections.

3.4 Evaluation and results

The quality of the computed token embeddings is tested when provided as input features to several text understanding tasks often evaluated in the literature (Zhou et al., 2016; Conneau et al., 2017; Ding et al., 2018). Their behaviour is examined and their usefulness is investigated on text understanding problems ranging from sentiment prediction and text classification to paraphrase detection, textual entailment recognition and implicit discourse relation classification. Therefore both sentence classification and sentence pair classification scenarios are being considered as it will be detailed further. In addition to this, all datasets used are analyzed, by presenting some of their characteristics as well as a discussion over the *"appropriateness"* of considering them for evaluation in the current setting, despite their typical employment in literature.

A comparison is provided between the proposed SATokE token embeddings and various context-independent word type embeddings: either pre-trained on large corpora on linear contexts (Mikolov et al., 2013b; Pennington et al., 2014) or pre-trained on syntactic dependency information (Levy and Goldberg, 2014). Additionally, the proposed token embeddings SATokE are compared to an alternative token embeddings method proposed in the literature (ELMo), which has been shown to provide improvements across multiple tasks (Peters et al., 2018). However, it is important to note that ELMo was trained on a large amount of data with a language modelling objective and additional character-level information. Also the resulting ELMo vectors are 1024-dimensional, as opposed to SATokE's 300-dimensional vectors. Results are provided using a fine-tuned version of ELMo in which embeddings are allowed to adapt during the end

task learning as well as a version in which the embeddings are fixed.

While more complex models can be employed for each individual task, the current work is exploratory and aims to assess the importance of using token embeddings compared to standard pre-trained word embeddings as well as the added value of using syntax on the different tasks. Furthermore, the architectures considered (recurrent and convolutional neural networks) are the building blocks of most methods employed in recent literature for tackling text understanding tasks, making them reasonable candidates to test the proposed token embeddings on.

3.4.1 Data appropriateness

Before deep diving into the evaluation of the datasets that provide a consensus in the literature for natural language understanding tasks assessment (Conneau et al., 2017; Conneau and Kiela, 2018; Zhou et al., 2016; Ding et al., 2018), it is important to note some characteristics that might make these datasets more or less appropriate for evaluation in the current setup.

Table 3.1 presents some data statistics of the considered datasets: average sentence length and percentage of sentences in the dataset for which a subject and/or an object relationship has not been detected by the parser. Based on the sentence length alone, these datasets can be split into two categories: datasets containing short sentences with an average sentence length of around 10 words (TREC, SICK and SNLI-20k) and datasets having longer sentences with an average sentence length around 20 words (MR, CR, SUBJ, SST-2 and MSRPC). In general one would expect the shorter sentences to be "easier" to parse and thus *"appropriate"* for evaluation in a setting that strongly relies on the availability and accuracy of parsed data. More precisely, by minimising the propagation of errors from the parsing stage into the token embeddings creation step, one could obtain more accurate SATokE representations.

Another important observation from Table 3.1 is that most datasets contain a high percentage of sentences in which dependency relations such as SUBJ or OBJ are not detected or do not exist. One such example from the MR dataset is presented in Table 3.3: the sentence considered lacks

	MR	CR	SUBJ	**TREC**	SST-2	**MSRPC**	**SICK**	**SNLI-20k**
average sentence length	22.34	20.61	25.17	**10.10**	19.89	22.24	**9.71**	**11.84**
%sent without SUBJ rel	21.00%	8.00%	10.00%	12.00%	21.00%	**2.00%**	6.00%	24.00%
%sent without OBJ rel	11.00%	15.00%	4.00%	25.00%	14.00%	**3.00%**	7.00%	10.00%

Table 3.1: Data characteristics. **Bold underlined** are the *"appropriate"* datasets

.

both a main verb and a SUBJ relation. Being part of a SUBJ and/or an OBJ relation provides important information about the role of a word in a sentence which can further influence the representations of the rest of the graph. Therefore having many sentences in a dataset for which such information is lacking or fails to be detected, can affect the performance of the representations obtained on the basis of the parse tree. Out of all the datasets considered, the one that has the lowest percentage of such sentences without a SUBJ or OBJ relation is MSRPC. This might indicate the *"appropriateness"* of using the MSRPC dataset for evaluation in the current setup despite its high average sentence length.

According to the above discussion, one could thus split the datasets according to whether they are adapted to syntactic analysis, in which case we refer to them as *"appropriate"* datasets, or not, in which case we refer to them as *"inappropriate"* datasets. The *"appropriateness"* of a dataset is therefore judged based on whether the sentences contained within it are well-formed and enable leveraging syntactic information or not, based on the criteria in Table 3.1. Having such a distinction should help assess the quality of the proposed representations while minimising the chances of propagating parsing errors. Therefore we further consider datasets TREC, MSRPC, SICK and SNLI-20k to fall under the *"appropriate"* datasets category and the remaining MR, CR, SUBJ and SST-2 to constitute *"inappropriate"* datasets.

3.4.2 Sentence understanding

The SATokE proposed token embeddings are evaluated in the following widely considered sentence understanding scenarios: sentiment analysis on movie reviews (MR, SST-2) and product

reviews (CR), text subjectivity/objectivity classification (SUBJ) and multi-class classification on question types (TREC). In the following the data used for the experiments is presented along with the specific implementation details pertaining to the used datasets, and the obtained results. It is important to note that out of the 5 datasets considered for assessing sentence understanding, only the TREC dataset seems to be *"appropriate"* for evaluation based on the analysis in Section 3.4.1.

Data

The **MR** dataset (Pang and Lee, 2005) is a movie reviews dataset with one sentence per review. The goal is to classify the reviews into two classes depending on the sentiment they express. The dataset is balanced with 50% of the reviews bearing positive sentiment and 50% negative. **SST-2** (Stanford Sentiment Treebank-2) (Socher et al., 2013b) is an extension of the MR dataset, with train / dev / test splits provided and having class distribution of 48% (negative) / 52% (positive). The **CR** dataset (Hu and Liu, 2004) contains customer reviews of various products (cameras, MP3s etc.) in which the goal is to determine the polarity of the reviews: 63% of the reviews in the dataset are positive and 36% negative.

SUBJ (Pang and Lee, 2004) is a balanced dataset containing subjective and objective sentences where the goal is to classify each sentence in one of the two classes. For the subjective sentences, 5000 movie review snippets (e.g., "bold, imaginative, and impossible to resist") were collected, while the objective data was collected from plot summaries. **TREC** (Li and Roth, 2002) is a question dataset that involves classifying a question into one of 6 question types, denoting whether it refers to a location, a person, a description, a numeric value, an abbreviation or an entity. The data is almost evenly distributed among the 6 classes: only the abbreviation class is underrepresented (0.01% of the data) and each of the rest of the classes is covered by approximately 20% of the cases. While for SST-2 and TREC the provided train/test splits are respected, for MR, CR and SUBJ, the reported results are based on 3-fold cross-validation.

A summary of the data statistics is presented in Table 3.2 and some sentence examples along

Corpus	Nb instances	Task	Nb classes	Split train/dev/test
MR	10.6k	sentiment classification (movie reviews)	2	3-fold CV
CR	3.7k	sentiment classification (product reviews)	2	3-fold CV
SUBJ	10.0k	subjectivity / objectivity classification	2	3-fold CV
TREC	5.9k	question type classification	6	4336 / 1116 / 500
SST-2	9.5k	sentiment classification (movie reviews)	2	6895 / 870 / 1819

Table 3.2: Data statistics for the sentence understanding datasets considered

Corpus	Example	Class
MR	"one from the heart."	positive
CR	"the little bag it comes with is cheap and useless."	negative
SUBJ	"bruce is a down on his luck tv news reporter."	objective
TREC	"what is the most popular sport in japan?"	entity
SST-2	"a vivid cinematic portrait."	positive

Table 3.3: Examples of sentences for the sentence understanding tasks considered

with their corresponding classes are presented in Table 3.3.

Implementation details

The hyper-parameters that were used to obtain the reported results on the end tasks are presented in Table 3.4. For each dataset, details are provided regarding the parameters involved in the computation of the token embeddings fed as input to each task as well as those required for the presented end task architectures. All parameters have been tuned on the corresponding development sets. The rest of the settings not covered in Table 3.4 (such as the value of the margin parameter γ, the batch size and the pre-processing steps for token embeddings computation) follow the general description in Section 3.3.2.

Figures 3.8 - 3.12 present the distribution of syntactic relations for the considered datasets: as explained in Section 3.3.2, only relations that have frequency higher than the threshold th_D are considered. All relations with frequency lower than th_D are merged into a single relation denoted "unknown_relation" also showed in the plots. After merging the low frequency relations into the "unknown_relation" generic tag, the resulting relation still has a low frequency in the case of the MR, SUBJ and SST-2 datasets and an important frequency in the case of the

Corpus	Token embeddings computation		LSTM runs			CNN runs			
	α_{LR}	rneg	α_{LR}	dp	nbh	α_{LR}	dp	nbf	fsz
MR	10^{-5}	1	10^{-5}	0.7	512	10^{-4}	0.7	256	3,4,5
CR	10^{-4}	5	10^{-5}	0.7	150	10^{-4}	0.8	256	3,4,5
SUBJ	10^{-5}	1	10^{-5}	0.7	512	10^{-4}	0.7	256	3,4,5
TREC	10^{-4}	10	10^{-5}	0.7	512	10^{-4}	0.8	256	3,4,5
SST	10^{-4}	5	10^{-5}	0.7	512	10^{-4}	0.7	128	3,4,5

Table 3.4: Best run parameters for each task: α_{LR} the initial learning rate, *rneg* the ratio of negative sampling, *dp* the dropout keep probability, *nbh* the number of hidden states, *nbf* the number of filters and *fsz* the filter sizes.

remaining two datasets CR and TREC. This indicates the number of low frequency relations is higher in the case of these two datasets. It further means that by merging them, one may lower the data sparsity at the cost of mixing potentially incompatible information.

The frequencies of the relations differ depending on the dataset, determining different levels of sparsity in the tensors. In general, it can be noticed that punctuation relations are significantly more frequent than the rest of the relations in 4 out of 5 datasets (except TREC). Additionally, by looking at the distributions one can notice a higher gap between different relation frequencies in the case of all datasets, except the *"appropriate"* one (TREC) for which the distribution of the relations seems to be more or less uniform. Having high variations in relation frequencies can affect the learning since it determines the amount of sampling done for each relation. This not only favours a better learning of relations that are more frequent, but also biases the learning of the least frequent ones: they have the same probability of being picked to constitute a part of a negative example as any other relation, yet they are more rarely sampled as part of positive points due to their low frequency. From this point of view one might expect the performance on the TREC dataset to be higher than on the other datasets.

Overall the most frequent relations seem to be: det (determiner), pobj (object of preposition), prep (prepositional modifier) and amod (adjectival modifier), with nsubj (nominal subject) following closely. Despite the high proportion of sentences without a subject relation observed in Section 3.4.1 for some of these datasets, the high frequency of the nsubj dependency relation

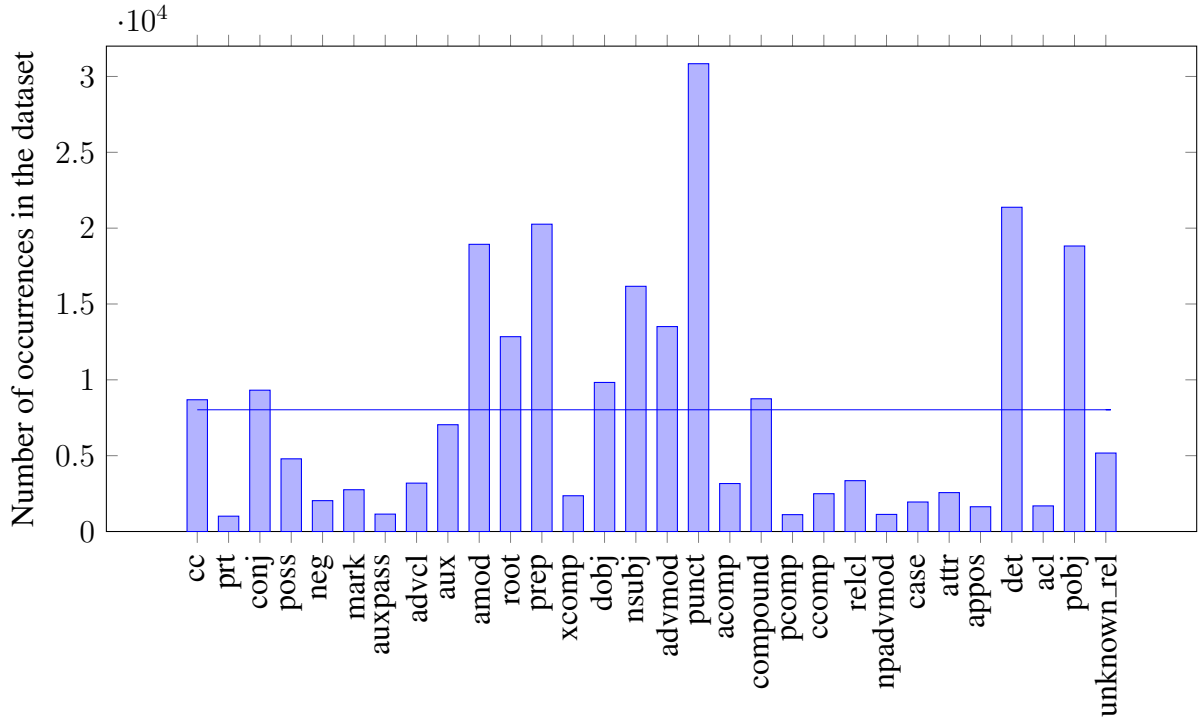


Figure 3.8: Distribution of the considered syntactic relations in the MR dataset.

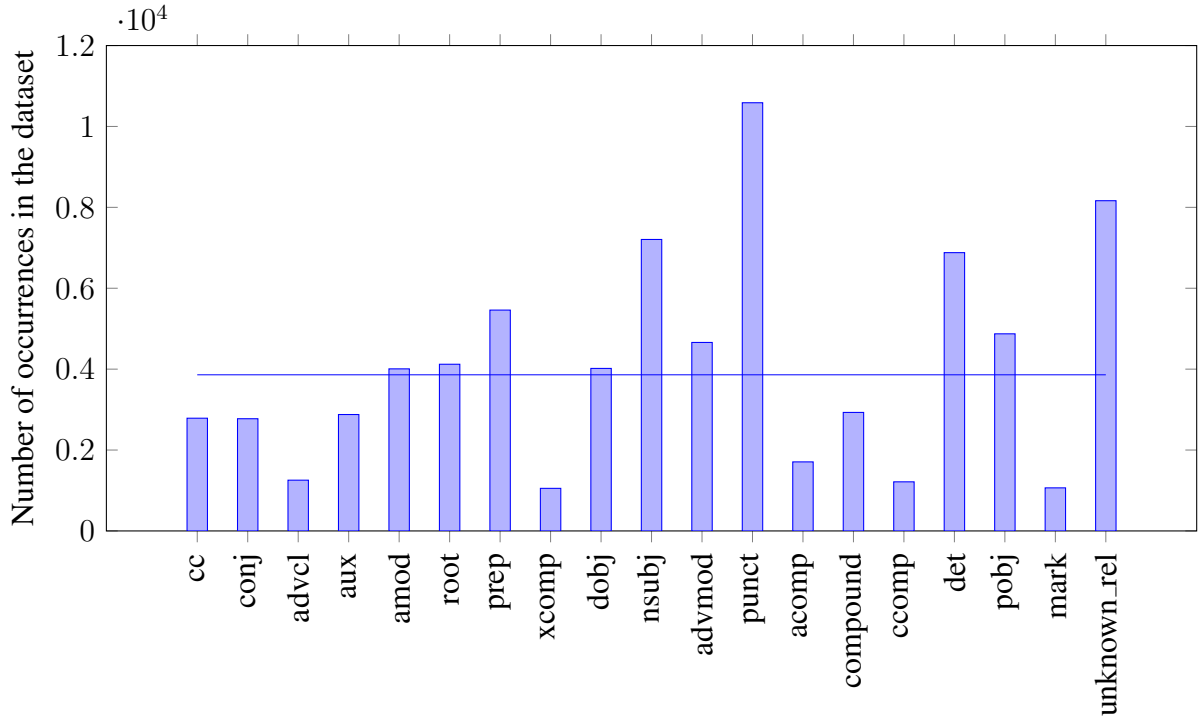


Figure 3.9: Distribution of the considered syntactic relations in the CR dataset.

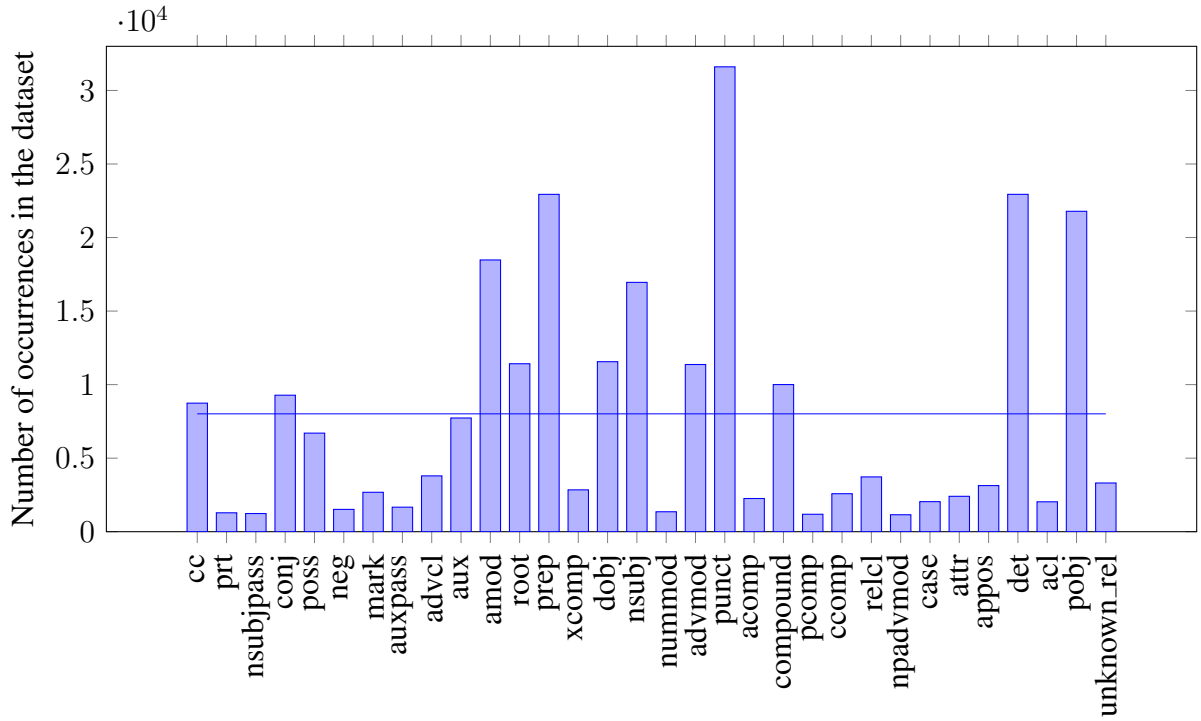


Figure 3.10: Distribution of the considered syntactic relations in the SUBJ dataset.

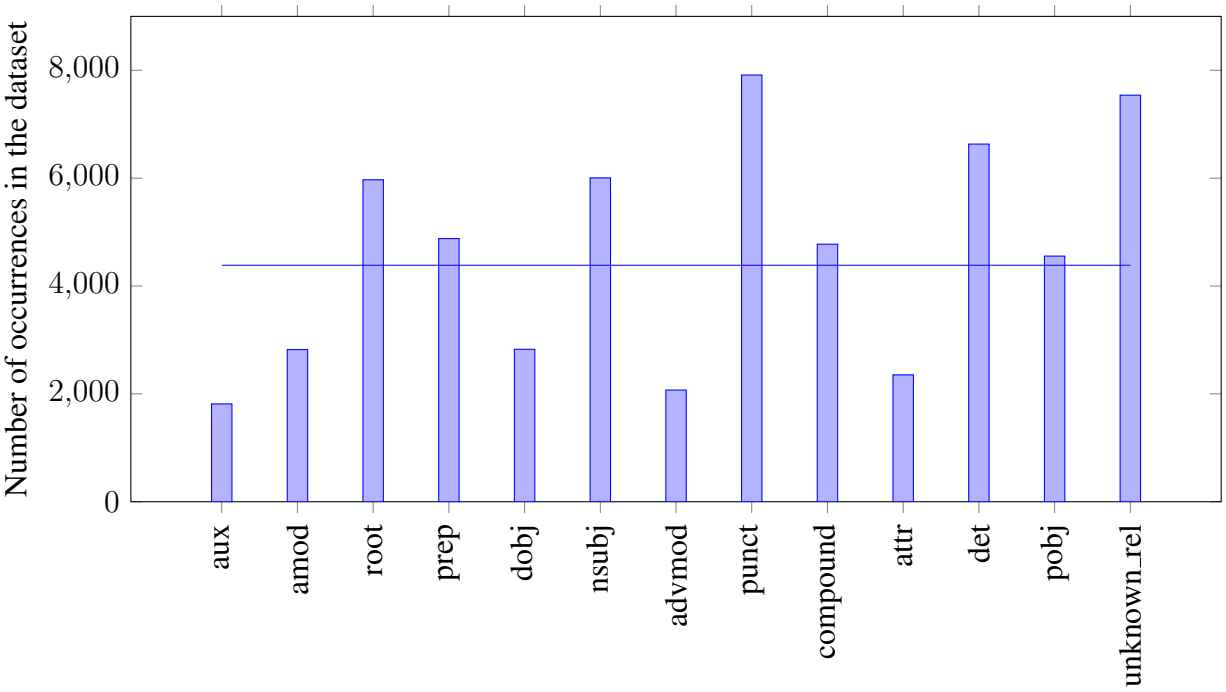


Figure 3.11: Distribution of the considered syntactic relations in the TREC dataset.

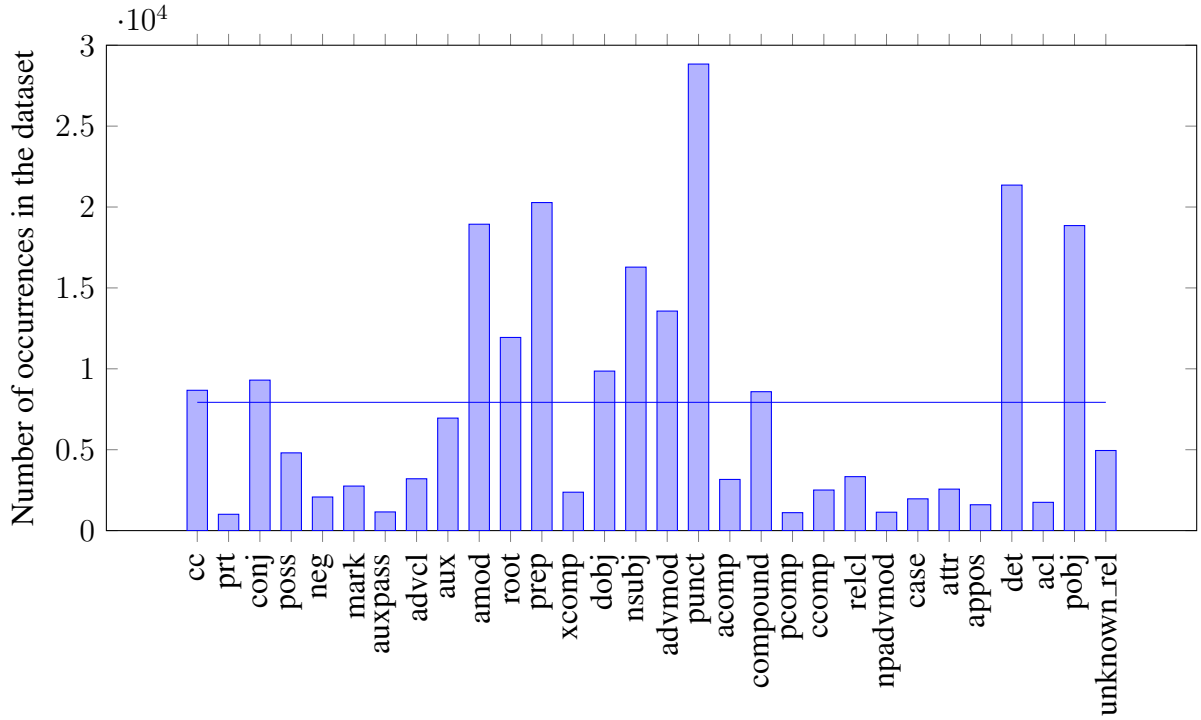


Figure 3.12: Distribution of the considered syntactic relations in the SST-2 dataset.

can be explained by the presence of multiple nsubj relations within the same sentence. More statistics related to the average number of subject relations detected per sentence, the proportion of sentences with more than 1 subject relation detected and the average dependency link distance are all presented in Table 3.5. The dataset with the lowest average of subject relations per sentence as well as the lowest percentage of sentences with more than 1 subject relationship and the lowest average dependency link distance is the TREC dataset. For all the others, more than half of the sentences containing a subject relationship actually have more than one such relationship detected, indicating complex compound structures.

Comparison to word type embeddings

Table 3.6 presents results of evaluating the SATokE proposed token embeddings on the sentence understanding tasks described previously. These are presented in comparison to results obtained when using pre-trained word type embeddings as input features for the same tasks. One can observe a consistent improvement in the performance of token embeddings over pre-trained

	MR	CR	SUBJ	**TREC**	SST-2
avg subj / sent	1.61	1.99	1.83	1.11	1.46
% sent / ds w/ $>$ 1 subj rel	48.56%	57.35%	56.60%	18.93%	43.49%
% sent / ds_subj w/ $>$ 1 subj rel	61.04%	62.31%	62,88%	21.31%	54.87%
avg link distance	3.23	3.28	3.27	2.51	3.1

Table 3.5: Elements of data complexity (from top to bottom): average number of subject relationships per sentence, percentage of sentences out of the whole dataset that have more than 1 subject relationship, percentage of sentences that have more than 1 subject relationship out of the sentences that have at least 1 subject relationship, average dependency link distance.

word type embeddings across all *"inappropriate"* datasets when using a CNN architecture and over the *"appropriate"* dataset when using both architectures. This result is invariant to the type of contexts used for pre-training the word type embeddings: token based representations outperform both word type embeddings trained on linear contexts (Pennington et al., 2014) as well as those trained on dependency contexts (Levy and Goldberg, 2014). Overall, the observed improvements confirm the importance of having a token-based representation.

In terms of architecture, CNN is preferred for almost all datasets when using the proposed token embeddings, while the opposite holds with word type embeddings: the preferred architecture for word type embeddings is the LSTM. This may be explained by the fact that the token embeddings already encode positional information by their construction using adjacency information, and thus they complement less the advantages of using an LSTM encoder.

Most improvements obtained by fine-tuning GloVe embeddings can be observed when using an LSTM architecture (datasets MR, SUBJ, SST, TREC) and only some hold for a CNN architecture as well (datasets MR and TREC). However, it is only in the case of the MR dataset that fine-tuning GloVe embeddings enables surpassing the score obtained using fixed SATokE embeddings. Overall no such tendency can be observed in the case of fine-tuning SATokE embeddings: since anyway the SATokE representations depend on the fine-tuned GloVe representations, this result is not surprising given the limited/no improvements brought about by the fine-tuning of the GloVe representations. Nevertheless, in general, SATokE surpasses GloVe embeddings on

Architecture / Embeddings	Datasets				
	"Inappropriate"				*"Appropriate"*
	MR	CR	SUBJ	SST-2	TREC
LSTM *Standard pre-trained Word Type embeddings*					
GloVe 300d (Pennington et al., 2014)	75.86	76.79	92.06	80.42	80.60
GloVe 300d (Pennington et al., 2014) (fine-tuned)	**77.24**	73.23	**92.78**	**81.80**	82.40
GloVe 300d + pos encod	73.27	74.42	90.80	79.60	81.20
GloVe 300d + self-attention	75.44	**77.37**	90.92	79.49	70.40
Dep-based WE 300d (Levy and Goldberg, 2014)	67.81	69.27	87.07	72.20	68.40
Dep-based WE 300d + pos encod	50.30	63.78	69.96	50.63	68.00
Dep-based WE 300d + self-attention	54.88	68.90	86.95	73.17	68.00
Current proposal Token embeddings					
SATokE 300d	74.39	73.00	91.19	79.60	**84.40**
SATokE 300d (fine-tuned)	72.13	66.91	89.99	76.58	54.20
CNN *Standard pre-trained Word Type embeddings*					
GloVe 300d (Pennington et al., 2014)	74.92	76.65	91.07	80.15	78.80
GloVe 300d (Pennington et al., 2014) (fine-tuned)	75.31	76.20	91.08	80.15	79.00
GloVe 300d + pos encod	72.53	74.34	90.79	78.33	82.40
GloVe 300d + self-attention	74.61	77.74	89.35	79.98	75.40
Dep-based WE 300d (Levy and Goldberg, 2014)	61.01	63.83	83.08	65.36	66.40
Dep-based WE 300d + pos encod	53.68	62.87	78.55	54.70	68.60
Dep-based WE 300d + self-attention	51.68	64.17	80.89	52.88	68.40
Current proposal Token embeddings					
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.60**
SATokE 300d (fine-tuned)	72.05	71.13	89.85	76.41	75.20

Table 3.6: Sentence classification results in terms of accuracy (%). Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using standard pre-trained word type embeddings: GloVe (Pennington et al., 2014) and Dep-based WE (Levy and Goldberg, 2014), including results using positional encodings and self-attention. All embeddings are fixed unless mentioned otherwise. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**.

the *"appropriate"* dataset and in all but two cases of the *"inappropriate"* data: when fine-tuning of GloVe embeddings is allowed, this yields slightly higher results than SATokE on the MR and SUBJ datasets.

As outlined in Table 3.6, the impact of using positional encodings and self-attention on top of word type embeddings is limited. Fixed positional encodings on top of word embeddings do not always help: one can only observe improvements on the TREC dataset when using GloVe embeddings with both LSTM and CNN architectures and a slight improvement when using dependency-based word embeddings as input to the CNN architecture. Nevertheless, these improvements are still not sufficient to exceed the accuracy provided by the proposed SATokE token representations.

For sentence classification, results using self-attention are mostly similar to those without attention, except for a few cases when self-attention helps: CR dataset when using GloVe embeddings with both architectures and dependency-based embeddings with CNN, SST-2 dataset with dependency-based word embeddings as input to LSTM and TREC dataset with a CNN architecture when using dependency-based embeddings. However, the obtained scores are still lower than those using the LSTM architecture on top of the same embeddings or any of the architectures with token embeddings.

Comparison to other token embeddings

As by construction token embeddings already encode information about the position and role in the sentence of the token, positional encodings and self-attention do not a priori bring an additional value on top of token embeddings. Therefore, since their improvements over type embeddings were limited, Table 3.7 focuses on comparing different token embeddings without positional encodings and self-attention: fixed ELMo (Peters et al., 2018), its fine-tuned version and the proposed SATokE embeddings. ELMo denotes a setting in which the layers of their deep bidirectional language model are all weighted equally (see Section 3.1) and the embeddings are fixed for the end task learning. ELMo (fine-tuned) stands for the original model in Peters et al.

Architecture / Token Embeddings	Datasets				
	"Inappropriate"				*"Appropriate"*
	MR	CR	SUBJ	SST-2	TREC
LSTM					
ELMo 1024d (Peters et al., 2018) (fine-tuned)	79.24	81.54	**94.27**	84.33	**94.00**
ELMo 1024d (Peters et al., 2018)	**79.43**	**82.36**	93.77	**84.38**	92.00
SATokE 300d	74.39	73.00	91.19	79.60	84.40
CNN					
ELMo 1024d (Peters et al., 2018) (fine-tuned)	**78.38**	82.10	93.14	83.01	**93.40**
ELMo 1024d (Peters et al., 2018)	78.03	**82.60**	**93.27**	**84.00**	92.60
SATokE 300d	76.72	78.81	91.73	82.13	92.60

Table 3.7: Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using existing token embeddings on sentence classification tasks: ELMo 1024d (Peters et al., 2018). All embeddings are fixed unless mentioned otherwise. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**.

(2018) that has been shown to provide useful representations for a variety of tasks and that allows fine-tuning of the weights of each layer along with end task learning.

The best results are obtained using ELMo (fixed) in three out of four cases (MR, CR and SST-2) of the *"inappropriate"* datasets and using ELMo (fine-tuned) for the remaining two datasets (SUBJ and TREC). From an architecture point of view, CNN seems to provide the best results only for the CR dataset (at a low margin) when using the ELMo embeddings, while all the other best results are obtained with an LSTM architecture on top of these embeddings. The fact that the ELMo model is implemented using a bidirectional LSTM may explain the compatibility of its output representations to an LSTM architecture rather than to a CNN one.

Obtaining better results with the ELMo embeddings than with SATokE might constitute a reflection of the nature of the datasets: MR, CR, SUBJ and SST-2 are all datasets of reviews in which the sentences have a varied vocabulary and not always a well-formed syntactic structure. This last aspect is important when dealing with methods that construct embeddings relying on accurate parsing (like in the case of SATokE). Thus, from a structural point of view, these datasets

can be more challenging: for example some sentences lack the SUBJ dependency and/or a main verb, as it can be seen from some of the examples in Table 3.3 and from the analysis in Section 3.4.1. Given the proposed SATokE token embeddings depend on a proper sentence structure, it is expected to observe a lower performance on such examples.

In addition to this, having a rich vocabulary favours methods trained on large amounts of data. Therefore, obtaining better scores using token embeddings trained with a language modelling objective is not a surprising result on these datasets. In particular in the case of reviews, the vocabulary employed can be often diverse and spelling errors can be present. To this end, the fact that ELMo leverages character-level embeddings can be useful to encode words that are rare or otherwise unknown. In contrast to that, SATokE does not leverage any a priori representation for unknown words [9]. Instead these words are replaced by a tag related to their part of speech and their representations are learned during the token embeddings computation. That is, the token embeddings are constrained to be close to a representation that is being constructed in the same process. If the proportion of unknown words in the dataset is high and, furthermore, if the dataset is small, performance can be affected in this process.

To validate this claim we can further look at the differences in performance between the results obtained using ELMo and the results obtained with the proposed token embeddings on the considered datasets. A parallel between these differences and the percentages of unknown words out of the vocabulary of each dataset is drawn in Table 3.8. One can observe a positive correlation between the two: the performance gap increases as the percentage of unknown words increases. While this does not necessarily imply some causality, it suggests a better handling of unknown words when constructing SATokE embeddings or even a concatenation to embeddings derived from character-level information (like in the case of ELMo) might yield improvements on the scores. We can observe that in the case of the *"appropriate"* dataset (TREC), the difference in performance between the SATokE proposed embeddings and ELMo is the lowest of all datasets. This may be because TREC contains only questions which are well-

[9] Unknown words are considered to be those that do not have a GloVe embedding or occur with a frequency lower than a threshold in the dataset, as explained in Section 3.3.2

Corpus	Performance Δ	% unknown words
TREC	1.40	3.04
SST-2	2.25	7.92
SUBJ	2.54	15.79
MR	2.71	15.49
CR	3.79	16.49

Table 3.8: Difference in performance between our token embeddings and ELMo (in absolute value of accuracy) and percentage of unknown words out of the vocabulary of each dataset.

formed sentences but also contains the lowest amount of unknown words. Based on a previous analysis, TREC is also the dataset that contains a rather uniform distribution of relations which may contribute to an efficient learning of the token embeddings for this dataset.

It is also worth noting that the SATokE token embeddings have been constructed by giving an equal weight to the semantic aspect (by constraining the embeddings to be close to GloVe) and the syntactic aspect (by forcing the embeddings to reconstruct the sentence parse tree). Therefore, it would be interesting to investigate whether embeddings constructed with less emphasis on the structure and more emphasis on the semantic aspect would perform better on these datasets. Also, an alternative approach would be to train relations embeddings on large datasets of reviews before using them to infer the token embeddings for the considered datasets. Additionally it is worth noting that the embeddings used in the ELMo settings have dimension 1024, which is 3 times the dimensionality of the proposed SATokE token embeddings.

3.4.3 Paraphrase detection

One task involving sentence pair understanding, which the proposed token representations can be evaluated on, is that of paraphrase detection. A *paraphrase* is defined as the restatement of a text such that its meaning is preserved. As it can be of use for several natural language processing tasks such as summarization, machine translation, plagiarism detection or question-answering, the task has received a lot of attention in the literature. Some work focuses on the input features to the task (Milajevs et al., 2014; Cheng and Kartsaklis, 2015; Issa et al., 2018),

while others explore architectures that can obtain competitive results (Socher et al., 2011a; He et al., 2015). The current work is part of the first category since the goal is to investigate the use of token representations for the task, as opposed to standard word type representations. Related work in the first category compares the use of different word type embeddings and composition methods applicable to the task (Milajevs et al., 2014), evaluates the use of abstract meaning representation parsing (Issa et al., 2018) or that of multi-sense word embeddings (Cheng and Kartsaklis, 2015). However, none of these investigates employing token embeddings for the task.

Data

The dataset used for the evalutation of the proposed SATokE token embeddings on paraphrase detection is the Microsoft Research Paraphrase Corpus (**MSRPC**) (Dolan and Brockett, 2005). It is a collection of 5801 sentence pairs extracted from news sources on the web and human-annotated for paraphrase / semantic equivalence, 65% of which are paraphrase pairs. The train (4076) / test (1725) split is provided. Some examples of sentences from MSRPC along with their corresponding classes are presented in Table 3.9.

According to the analysis in Section 3.4.1, MSRPC has the characteristics of a dataset that is *"appropriate"* for evaluation in the current setting, as almost all of its sentences are well-formed, having at least one subject and one object dependency relation. However, MSRPC is one of the datasets with the highest average sentence length out of all analyzed datasets.

Implementation details

The parameters used to obtain the reported results for the task of paraphrase detection are presented in Table 3.10. Information is included regarding the parameters involved in the computation of the token embeddings as well as those used in the end task model for paraphrase prediction. The parameters not included in Table 3.10 follow the description and values reported

Example	Class
S1: "His mother contacted the federal public defender's office in Sacramento, which has agreed to handle his surrender, she says."; S2: "His family approached the federal defender's office in Sacramento about arranging his surrender."	paraphrase
S1: "July 1st is the sixth anniversary of hong kong 's return to chinese rule."; S2: "The rally overshadowed ceremonies marking the sixth anniversary of hong kong's return to china on 1 july 1997."	not paraphrase
S1: "It added that unless the problems are fixed, 'the scene is set for another accident.' "; S2: "Without reform, 'the scene is set for another accident', the report warned."	paraphrase

Table 3.9: Examples of sentences from the MSRPC dataset.

Corpus	Token embeddings computation		LSTM runs			CNN runs			
	α_{LR}	rneg	α_{LR}	dp	nbh	α_{LR}	dp	nbf	fsz
MSRPC	10^{-4}	1	10^{-5}	0.7	150	10^{-4}	0.7	128	3,4,5

Table 3.10: Best run parameters for the task of paraphrase detection: α_{LR} the initial learning rate, *rneg* the ratio of negative sampling, *dp* the dropout keep probability, *nbh* the number of hidden states, *nbf* the number of filters and *fsz* the filter sizes.

in Section 3.3.2.

Figure 3.13 presents the distribution of syntactic relations in the MSRPC corpus. As it was the case for many datasets analysed in Section 3.4.2, there is a high variation between the frequencies of different relations, potentially making the learning more challenging. The most frequent relations in the case of MSRPC are: punct (punctuation), prep (prepositional modifier), pobj (object of preposition), compound and det (determiner).

	MSRPC
avg subj / sent	1.83
% sent / ds w/ $>$ 1 subj rel	58.68%
% sent / ds_subj w/ $>$ 1 subj rel	59.79%
avg link distance	3.2

Table 3.11: Elements of data complexity (from top to bottom): average number of subject relationships per sentence, percentage of sentences out of the whole dataset that have more than 1 subject relationship, percentage of sentences that have more than 1 subject relationship out of the sentences that have at least 1 subject relationship, average dependency link distance.

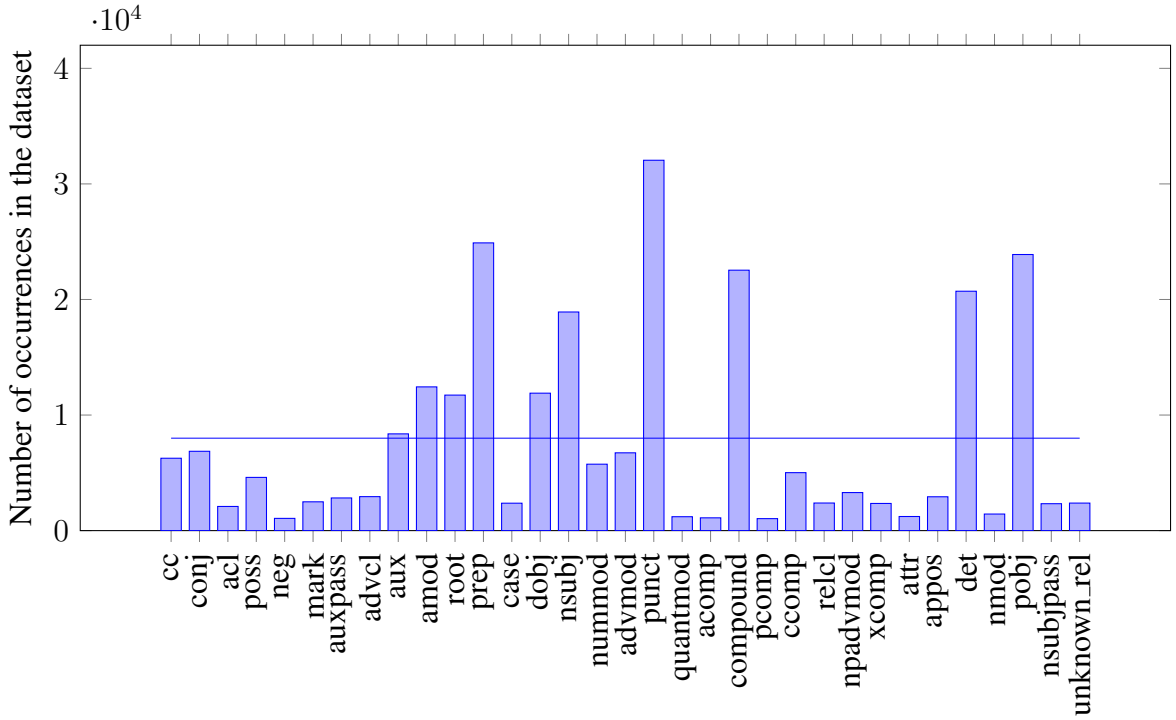


Figure 3.13: Distribution of the considered syntactic relations in the MSRPC dataset.

The high frequency of the nsubj (nominal subject) dependency relation is explained by the presence of a high number of sentences for which more than 1 subject relation was detected. Table 3.11 shows that almost 60% of the sentences that have a subject relation detected, actually have more than one such relation, which indicates the presence of complex structures. Additionally the average dependency link distance of 3.2 is among the highest of the analyzed datasets. Therefore, although *"appropriate"* for analysis in the current setup, the MSRPC dataset represents a challenging dataset due to its characteristics.

Comparison to word type embeddings

Table 3.12 presents the results of using word type embeddings as input to the task of paraphrase detection in comparison to using the proposed token embeddings. One can observe an improvement in the performance of token embeddings over both GloVe and dependency-based pre-trained word type embeddings regardless of the chosen architecture. The best overall results are however obtained using the proposed SATokE token embeddings fed as input to the CNN

architecture. Fine-tuning GloVe embeddings does not yield improved results over SATokE in either of the two architectures.

In contrast to the sentence understanding tasks results in Section 3.4.2, the dependency-based word type embeddings provide scores almost as high as the GloVe embeddings. This could be due to the fact that sentences in MSRPC provide rich syntactic structures as compared to the sentences considered in the sentence understanding tasks. Thus dependency information may be leveraged more for such examples than for sentences that lack such properties. The MSRPC corpus also contains the lowest percentage of sentences without a SUBJ or OBJ relation (out of the analyzed corpora) as outlined in Section 3.4.1.

Positional encodings provide a marginal improvement (0.95%) when using the GloVe embeddings and have the opposite effect when using dependency-based word embeddings. Self-attention yields slightly better scores on top of GloVe embeddings when using an LSTM architecture with an increase of 1.6% in performance. For the dependency-based word embeddings self-attention does not provide improvements. However, no scores computed using positional encodings or self-attention on top of pre-trained word type embeddings outperform the proposed token embeddings. This result confirms the benefits of using contextualized representations for the task considered.

Comparison to token embeddings methods

Table 3.13 compares results obtained using the proposed SATokE token embeddings to those obtained using the ELMo token embeddings from Peters et al. (2018). The two settings ELMo (fine-tuned) and ELMo correspond to two settings: the original setting in which a weighting of the layers of the deep bidirectional language model is learned along with the end task (as described in Section 3.1) and a version in which the layers are weighted equally and the embeddings constitute a fixed input to the task.

One can observe that the proposed SATokE token embeddings outperform both versions of

Architecture / Embeddings	*"Appropriate"* datasets
	MSRPC
LSTM	
Standard pre-trained Word Type embeddings	
GloVe 300d (Pennington et al., 2014)	68.63
GloVe 300d (Pennington et al., 2014) (fine-tuned)	68.28
GloVe 300d + pos encod	**69.50**
GloVe 300d + self-attention	68.17
Dep-based WE 300d (Levy and Goldberg, 2014)	67.88
Dep-based WE 300d + pos encod	66.66
Dep-based WE 300d + self-attention	67.47
Current proposal Token embeddings	
SATokE 300d	69.27
SATokE 300d (fine-tuned)	69.15
CNN	
Standard pre-trained Word Type embeddings	
GloVe 300d (Pennington et al., 2014)	66.89
GloVe 300d (Pennington et al., 2014) (fine-tuned)	65.25
GloVe 300d + pos encod	67.18
GloVe 300d + self-attention	68.52
Dep-based WE 300d (Levy and Goldberg, 2014)	66.26
Dep-based WE 300d + pos encod	65.85
Dep-based WE 300d + self-attention	67.65
Current proposal Token embeddings	
SATokE 300d	**70.32**
SATokE 300d (fine-tuned)	69.44

Table 3.12: Sentence pair classification results on the task of paraphrase detection in terms of accuracy (%). Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using standard pre-trained word type embeddings: GloVe (Pennington et al., 2014) and Dep-based WE (Levy and Goldberg, 2014), including results using positional encodings and self-attention. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**

.

Architecture / Token Embeddings	*"Appropriate"* datasets
	MSRPC
LSTM	
ELMo 1024d (Peters et al., 2018) (fine-tuned)	69.10
ELMo 1024d (Peters et al., 2018)	68.50
SATokE 300d	**69.27**
CNN	
ELMo 1024d (Peters et al., 2018) (fine-tuned)	66.20
ELMo 1024d (Peters et al., 2018)	66.84
SATokE 300d	**70.32**

Table 3.13: Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using existing token embeddings on the task of paraphrase detection: ELMo 1024d (Peters et al., 2018). All embeddings are fixed unless mentioned otherwise. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**.

ELMo, despite the fact that they use less parameters: 300-dimensional representations for the proposed token embeddings vs 1024-dimensional representations for ELMo. It is also important to note that ELMo was trained on large amounts of data, while the proposed SATokE token embeddings are computed only on the basis of the corpus at hand (leveraging however information from pre-trained word type embeddings). The second best results are, not surprisingly, obtained by the version of ELMo that allows fine-tuning of the embeddings on the end task (when provided as input to an LSTM). In terms of architecture, as previously, the best results obtained by SATokE are when using a CNN, while the best ELMo results are obtained with LSTM.

Similarly to some of the previously considered datasets for sentence understanding in Section 3.4.2, the MSRPC dataset contains challenging and complex examples, having among the highest average sentence length and average dependency link distance out of all datasets. However, with dependency relations such as SUBJ, OBJ and MOD being present in most of the examples, the proposed SATokE token embeddings efficiently learn encodings of the parse trees of the sentences, leveraging the syntactic structure. This thus outlines the benefit of using the

syntactically-informed token embeddings on such sentences.

3.4.4 Textual entailment recognition

Another task widely studied in the literature that pertains to sentence pair understanding and that has applicability in many fields is that of textual entailment recognition (Dagan et al., 2006; Mirkin et al., 2009; Rocktäschel et al., 2016; Chen et al., 2017; Peters et al., 2018). This is based on the idea that the same meaning can be expressed by, or inferred from, different text fragments and thus having an accurate model for textual entailment could benefit many natural language application such as question-answering, machine translation or summarization.

Textual entailment is defined as the directional relationship that holds between two text fragments T and H, called the entailing *Text* and the entailed *Hypothesis* respectively, if by reading T a human would infer that H is most likely true (Dagan et al., 2006). This definition assumes both common human understanding and background knowledge, which underlines the challenges arisen by this task. Much work has focused on textual entailment recognition throughout time with only limited literature particularly investigating the use of different word representations as input to the task (Lan and Jiang, 2018; Peters et al., 2018). The current proposal falls in this category of work by investigating the contribution of the proposed syntactically-aware token representations to detecting textual entailment relations.

Data

To evaluate the proposed token embeddings on the task of textual entailment recognition, two standard datasets often evaluated in literature are used: SICK (Sentences Involving Compositional Knowledge) (Bentivogli et al., 2016) and SNLI (Stanford Natural Language Inference) (Bowman et al., 2015). Table 3.14 presents examples of sentence pairs from both datasets along with their corresponding labels.

Examples - Corpus	Class
SICK	
T: "An old woman is shaking hands with a man."; H: "Two persons are shaking hands."	entailment
T: "A baby is licking a dog."; H: "A dog is licking a baby."	neutral
T: "A squirrel is lying down."; H: "A squirrel is running around in circles."	contradiction
SNLI	
T: "A man looking over a bicycle's rear wheel in the maintenance garage with various tools visible in the background." ; H: "A person is in a garage."	entailment
T: "A couple walk hand in hand down a street." ; H: "The couple is married."	neutral
T: "A person dressed in a dress with flowers and a stuffed bee attached to it, is pushing a baby stroller down the street." ; H: "A lady sitting on a bench in the park."	contradiction

Table 3.14: Examples of sentence pairs used for the evaluation of textual entailment recognition.

SICK (Sentences Involving Compositional Knowledge) (Bentivogli et al., 2016) is a corpus of 10k sentence pairs, drawn from image and video descriptions and split into: entailment, neutral and contradiction, with a train/dev/test split provided.

SNLI (Stanford Natural Language Inference) (Bowman et al., 2015) represents a benchmark for the evaluation of systems on the task of textual entailment. It consists of 570k human-written English sentence pairs labelled as entailment, contradiction or neutral and divided into pre-defined train, development and test sets. In the current work, the development (20k) set is used as a train set and the provided test (20k) set is kept for testing. This split is further referred to as **SNLI-20k**.

According to the analysis in Section 3.4.1, both these datasets are *"appropriate"*: they are adapted to syntactic analysis due to their low average sentence length. In addition to this, the SICK dataset also has the property of having a low percentage of sentences without certain dependency relations, unlike the case of SNLI-20k for which a high number of sentences do not contain a SUBJ dependency relation.

Corpus	Token embeddings computation		LSTM runs			CNN runs			
	α_{LR}	rneg	α_{LR}	dp	nbh	α_{LR}	dp	nbf	fsz
SICK	10^{-4}	1	10^{-5}	0.7	512	10^{-4}	0.7	128	3,4,5
SNLI-20k	10^{-4}	5	10^{-5}	0.7	150	10^{-4}	0.8	256	3,4,5

Table 3.15: Best run parameters for the task of textual entailment recognition: α_{LR} the initial learning rate, *rneg* the ratio of negative sampling, *dp* the dropout keep probability, *nbh* the number of hidden states, *nbf* the number of filters and *fsz* the filter sizes.

Implementation details

The parameters used for the reported results are provided in Table 3.15. As previously, the parameters that are not included in Table 3.15, namely the value of the margin parameter γ, the batch size and all the parameters regarding the pre-processing steps for token embeddins computation, follow the description in Section 3.3.2.

Figure 3.14 and Figure 3.15 present the distribution of the frequency of syntactic relations considered for these datasets. The distribution of the relations frequencies shows a high variation in the case of the SNLI-20k dataset, while the SICK dataset presents a more balanced situation with almost half of the relations having a frequency above average and the rest below. For both datasets the frequency of the "unknown_rel" relation is low with respect to other relations, indicating a not very diverse set of infrequent relations. The most frequent relations are: det (determiner), pobj (object of preposition), prep (preposition), punct (punctuation) and amod (adjectival modifier) for both datasets, with aux (auxiliary) and nsubj (nominal subject) being additionally well represented in SICK.

Table 3.16 outlines further characteristics about the data: SICK and SNLI-20k are the datasets with the lowest average number of subject relations per sentence and have low percentages of sentences with more than 1 subject relation detected (around 9.6% for SICK and 14.33% for SNLI-20k). The average dependency link distance is also slightly lower than in the case of the previously analyzed datasets.

	SICK	SNLI-20k
avg subj / sent	1.04	0.89
% sent / ds w/ $>$ 1 subj rel	9.06%	11.02%
% sent / ds_subj w/ $>$ 1 subj rel	9.60%	14.33%
avg link distance	1.95	2.44

Table 3.16: Elements of data complexity (from top to bottom): average number of subject relationships per sentence, percentage of sentences out of the whole dataset that have more than 1 subject relationship, percentage of sentences that have more than 1 subject relationship out of the sentences that have at least 1 subject relationship, average dependency link distance.

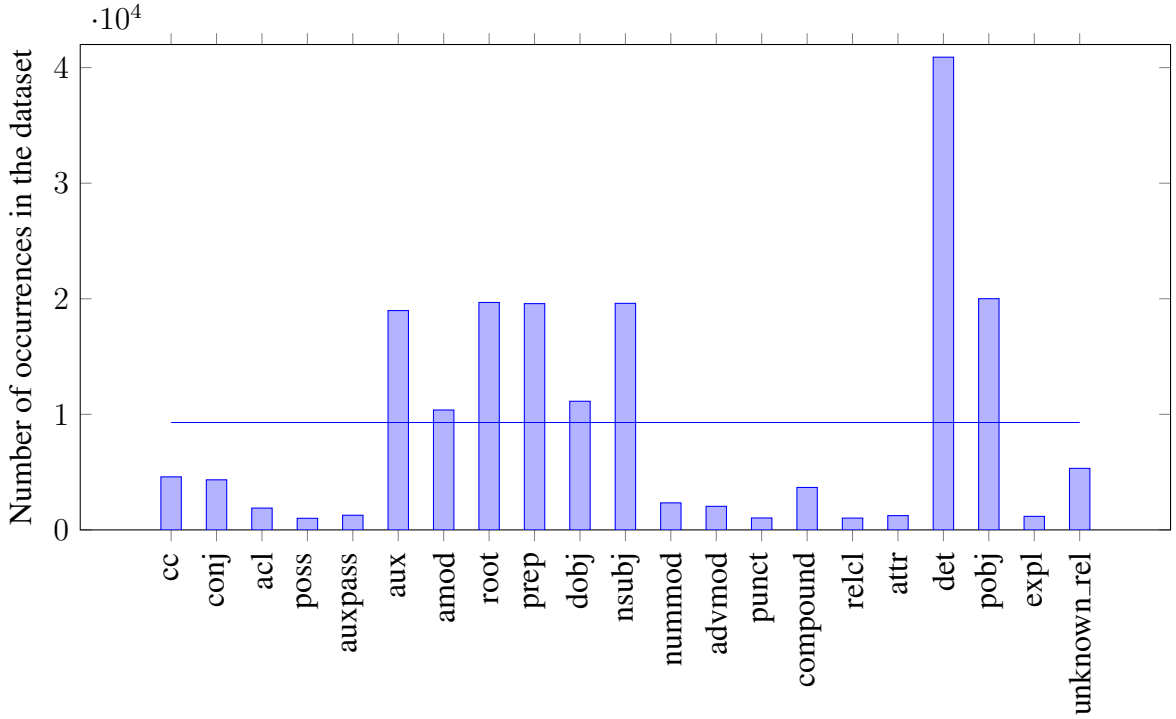


Figure 3.14: Distribution of the considered syntactic relations in the SICK dataset.

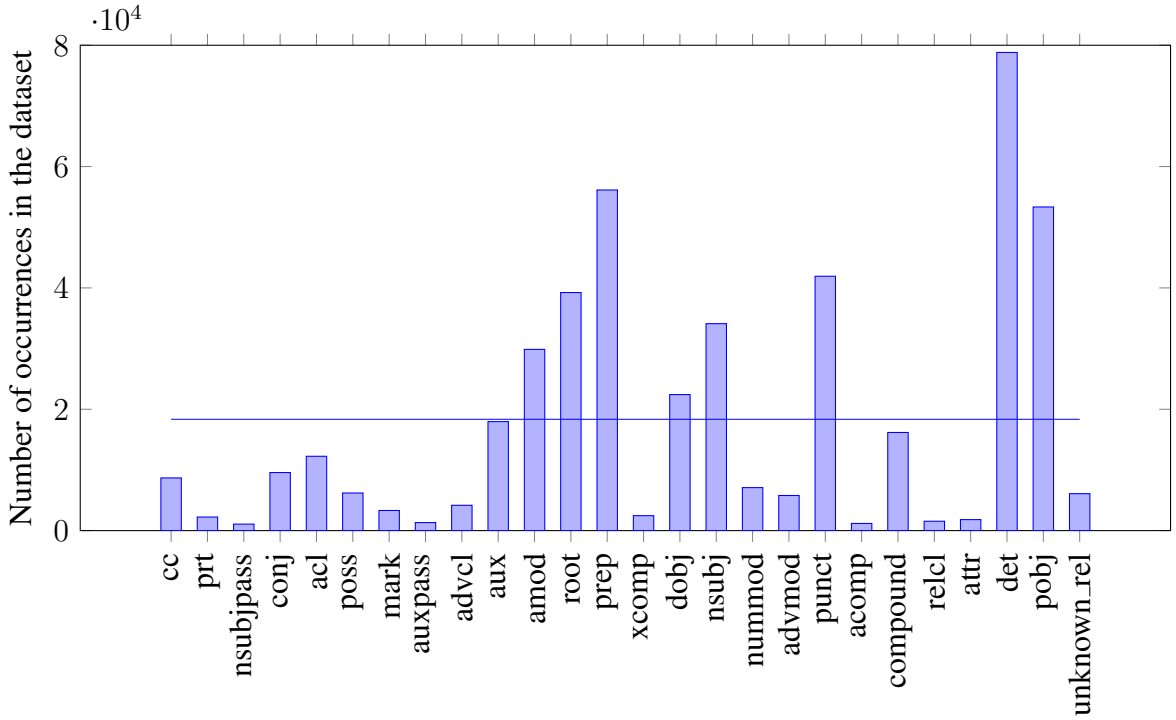


Figure 3.15: Distribution of the considered syntactic relations in the SNLI-20k dataset.

Comparison to word type embeddings

The results of using word type embeddings and the proposed SATokE token embeddings as input to the task of textual entailment recognition are presented in Table 3.17. For both datasets, SICK and SNLI-20k, feeding the SATokE token embeddings as input features yields better results than using word type embeddings. Even fine-tuning the GloVe embeddings with respect to the end task only results in a slight improvement on the SNLI-20k dataset when used within an LSTM architecture and yields no improvements within a CNN architecture. The proposed SATokE token embeddings outperform both GloVe and dependency-based word type embeddings and obtain the highest scores when input to an LSTM architecture in the case of SICK and to a CNN architecture for the SNLI-20k dataset.

The difference in performance between the GloVe and the dependency-based word type embeddings is lower in the case of the SICK dataset than in the case of the SNLI-20k dataset. Similarly to the observations in Section 3.4.3, this could be due to the different characteristics of the sentences in the two datasets: SICK contains a less diverse vocabulary than SNLI-20k,

yet at the same time the sentences in SICK are more challenging from a structural point of view as will be later seen in Table 3.24. This enables dependency-related information to be leveraged and prevents successful results just on the basis of word overlap statistics.

Using positional encodings only brings a marginal improvement (0.25%) in the case of dependency-based word type embeddings with an LSTM architecture on the SICK dataset. The results obtained though are still lower than those when using a CNN architecture with the same embeddings but without positional encodings. In all the other cases positional encodings decrease the scores on both datasets using both architectures.

Self-attention improves the scores for both datasets when using a CNN architecture on top of GloVe embeddings: 1.5% for SICK and 0.7% for SNLI-20k. However for both datasets the obtained scores remain lower than the best results obtained with the same embeddings (without positional encoding) when input to an LSTM network. A marginal improvement is also observed when using dependency-based word type embeddings as input to the LSTM architecture for the SICK dataset. However, similarly, the improvement is not sufficient to surpass the results obtained by the dependency-based word type embeddings when input to a CNN architecture without using self-attention.

None of the improvements present due to positional encodings or self-attention is sufficient to outperform the accuracy provided by the proposed SATokE token representations.

Comparison to token embeddings methods

In Table 3.18, the proposed token embeddings are compared to the ELMo token embeddings introduced in Peters et al. (2018). As previously, two settings are considered: allowing the weights of the intermediate layers obtained from the deep bidirectional language model to tune during the task learning (cf. Section 3.1) or weighting all layers equally and fixing the resulting embeddings for the end task learning. Since token embeddings already include positional information and since positional encodings and self-attention did not result in improvements over

Architecture / Embeddings	*"Appropriate"* datasets	
	SICK	SNLI-20k
LSTM *Standard pre-trained Word Type embeddings*		
GloVe 300d (Pennington et al., 2014)	61.96	57.13
GloVe 300d (Pennington et al., 2014) (fine-tuned)	60.33	**57.51**
GloVe 300d + pos encod	61.35	55.74
GloVe 300d + self-attention	58.76	50.62
Dep-based WE 300d (Levy and Goldberg, 2014)	59.88	50.55
Dep-based WE 300d + pos encod	60.13	33.53
Dep-based WE 300d + self-attention	60.59	47.79
Current proposal Token embeddings		
SATokE 300d	**63.59**	56.76
SATokE 300d (fine-tuned)	59.68	51.51
CNN *Standard pre-trained Word Type embeddings*		
GloVe 300d (Pennington et al., 2014)	60.23	55.84
GloVe 300d (Pennington et al., 2014) (fine-tuned)	58.11	55.48
GloVe 300d + pos encod	59.78	55.12
GloVe 300d + self-attention	61.74	56.48
Dep-based WE 300d (Levy and Goldberg, 2014)	60.76	47.84
Dep-based WE 300d + pos encod	59.11	45.34
Dep-based WE 300d + self-attention	59.60	46.38
Current proposal Token embeddings		
SATokE 300d	**62.53**	**57.72**
SATokE 300d (fine-tuned)	59.11	53.97

Table 3.17: Sentence pair classification results on the task of textual entailment recognition in terms of accuracy (%). Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using standard pre-trained word type embeddings: GloVe (Pennington et al., 2014) and Dep-based WE (Levy and Goldberg, 2014), including results using positional encodings and self-attention. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**

.

word type embeddings, such experiments were not considered along with token embeddings.

The best results are obtained using the SATokE proposed token embeddings for the SICK dataset and using the fine-tuned ELMo token embeddings on the SNLI-20k dataset. In both cases, LSTM is the preferred architecture, but considering the SATokE embeddings outperforms the results obtained with ELMo when input to a CNN.

Similarly to previous observations in Sections 3.4.2 and 3.4.3, the results can be analyzed with respect to the characteristics of the datasets. The sentences in the SICK dataset have a less diverse vocabulary than the ones in SNLI-20k. However, they pose a structural challenge in that understanding the relation between two sentences can not be efficiently achieved just by looking at the words themselves, but requires an understanding of the compositional aspect of the sentences, like shown in the examples in Table 3.14 for the SICK dataset. Thus the proposed token embeddings that explicitly leverage information about the structures of the sentences from a dependency point of view, are particularly useful in this case. At the same time having a rich vocabulary favours methods trained on large amounts of data that can additionally leverage character-level information. This can be seen in the case of SNLI-20k: ELMo embeddings pre-trained on the 1B Word Benchmark (Chelba et al., 2013) obtain the best results on this dataset.

3.4.5 Further analysis on sentence understanding tasks

In the following a further look is provided into the tasks already considered and the focus shifts on different aspects concerning the results. A first look is aimed at comparing the proposed SATokE token embeddings to the 256-dimensional ELMo token embeddings. Then, a discussion is provided around the comparison of SATokE to general purpose word type embeddings computed on dependency contexts (Levy and Goldberg, 2014).

Further, one ablation study analyzes the impact of using only adjacency information compared to adding the syntactic information when constructing the token embeddings. Two other sets

Architecture / Token Embeddings	*"Appropriate"* datasets	
	SICK	SNLI-20k
LSTM		
ELMo 1024d (Peters et al., 2017) (fine-tuned)	61.63	**<u>60.64</u>**
ELMo 1024d (Peters et al., 2017)	61.85	59.15
SATokE 300d	**<u>63.59</u>**	56.76
CNN		
ELMo 1024d (Peters et al., 2017) (fine-tuned)	57.88	56.62
ELMo 1024d (Peters et al., 2017)	60.31	56.40
SATokE 300d	**62.53**	**57.72**

Table 3.18: Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300d) to results obtained using existing token embeddings on the textual entailment recognition task: ELMo 1024d (Peters et al., 2018). All embeddings are fixed unless mentioned otherwise. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **<u>bold underlined</u>**.

of experiments analyze the impact of having fine-grained punctuation marks and fine-grained dependency relations. Then the results obtained using token embeddings computed in a 2-step setup are analyzed and finally a qualitative analysis is also performed on the considered tasks.

Comparison to 256-dimensional ELMo token embeddings

The results previously presented in Table 3.7, Table 3.13 and Table 3.18 are obtained using 1024-dimensional ELMo embeddings, while the proposed SATokE embeddings are 300-dimensional - to enable the compatibility to the 300-dimensional GloVe representations on which they are constrained. Thus, to provide a fair comparison to embeddings of similar dimensions, Table 3.19 compares SATokE to 256-dimensional ELMo token embeddings. The results show that, given similar dimensionality, SATokE outperforms ELMo across all but one dataset (SUBJ), regardless of whether they are labeled as *"appropriate"* or *"innapropriate"* and even despite fine-tuning the ELMo embeddings.

The significant drop in performance observed with respect to previous results using ELMo

Architecture / Embeddings	Datasets							
	"Inappropriate"				*"Appropriate"*			
	MR	CR	SUBJ	SST-2	TREC	MSRPC	SICK	SNLI-20k
LSTM								
ELMo 256d	**74.93**	**77.32**	**92.20**	**80.64**	**91.00**	64.00	57.37	42.44
ELMo 256d (fine-tuned)	73.72	77.29	91.93	78.06	89.80	64.46	57.78	40.63
SATokE 300d	74.39	73.00	91.19	79.60	84.40	**69.27**	**63.59**	**56.76**
CNN								
ELMo 256d	74.80	78.38	91.99	80.37	90.6	65.44	50.48	41.96
ELMo 256d (fine-tuned)	72.53	77.56	91.14	78.83	89.8	66.43	51.18	40.98
SATokE 300d	**76.72**	**78.81**	91.73	**82.13**	**92.6**	**70.32**	**62.53**	**57.72**

Table 3.19: Comparison of the results obtained using the proposed syntactically-aware token embeddings (SATokE 300-dimensional) to results obtained using the 256-dimensional ELMo token embeddings (Peters et al., 2018) across all tasks considered in Sections 3.4.2, 3.4.3, 3.4.4. All embeddings are fixed unless mentioned otherwise. Best results for each architecture (LSTM, CNN) are in **bold**. Best overall results across architectures are in **bold underlined**.

1024d suggest that the superior results of ELMo 1024d are (also) linked to its increased dimensionality compared to ELMo 256d, which allows it to retain useful information. It is important to note that ELMo 256d fails to capture the same amount and quality of information as SATokE, despite it being trained on the same amount of data as ELMo 1024d, which is far larger than the amount of data used in the case of SATokE.

With respect to the architecture type, CNN continues to be preferred for SATokE 300d, except in the case of the SICK dataset when the best results are obtained using an LSTM. For ELMo 256d the best results are obtained using CNN for the CR and MSRPC datasets and using LSTM for all the others. Additionally it can be observed that the difference between scores obtained by SATokE and ELMo 256d on the *"appropriate"* datasets is higher (between 1.6% and 17%) than that present on the *"inappropriate"* datasets (no more than 2%), suggesting SATokE clearly yields improvements when *"appropriate"* data is provided.

Linear vs dependency contexts

The word type embeddings trained on dependency contexts (Levy and Goldberg, 2014) yield consistently worse results than GloVe embeddings in the current evaluation, with positional encodings providing an even further performance decrease. This result is consistent with the analysis in Ghannay et al. (2016) who show that embeddings trained on dependency contexts obtain lower results compared to other embeddings on semantic tasks such as analogical reasoning and similarity tasks, despite their high performance on named entity recognition (NER), part-of-speech tagging or chuncking. However, contrary to a first interpretation, this does not imply that dependency information is not useful for semantic tasks: it can be seen that the induced SATokE token representations based strongly on dependency information obtain competitive scores. This could lead to the belief that the value of using dependency information is higher when this information is injected into the token representations directly, by using knowledge from the parse tree of the sentence, than when used as context for creating generic word type embeddings from a large corpus.

Furthermore, it has been observed that the difference in performance between GloVe and Dep-based WE is considerably smaller for some datasets than for others: only 1.7% difference in performance for MSRPC and 1.2% for SICK. In contrast to this, the score differences for all the other datasets are between 6.5% and 14%: 8% for MR, 7.5% for CR, 5.3% for SUBJ, 14% for TREC, 7% for SST-2, 6.5% for SNLI-20k.

Interestingly, the datasets on which word type embeddings trained on dependency contexts provide results almost as good as GloVe are the same on which the proposed SATokE token embeddings outperform ELMo, namely MSRPC and SICK. This can further support the observations regarding the link between differently constructed embeddings and the characteristics of the sentences in these datasets as explained in Section 3.4.3 and Section 3.4.4: the MSRPC and SICK datasets contain syntactically-rich structures and thus benefit more from having a contextualized representation that takes the structure of the sentence into account. Also, according to the analysis in Section 3.4.1, they have the lowest percentages of sentences without SUBJ and

Architecture / Embeddings	Datasets							
	"Inappropriate"				*"Appropriate"*			
	MR	CR	SUBJ	SST-2	TREC	MSRPC	SICK	SNLI-20k
CNN								
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.60**	**70.32**	**62.53**	**57.72**
SATokE 300d only-adjacency	50.67	63.62	50.94	51.45	33.20	66.43	56.50	36.79

Table 3.20: Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the proposed syntactically-aware token embeddings (SATokE 300d) to token embeddings computed using only adjacency information from the sentence graph (SATokE 300d only-adjacency). All embeddings are fixed. Best results are in **bold**.

without OBJ dependency relations.

Ablation study - impact of syntax

In order to assess the contribution of syntax, an additional experiment is performed by constructing token embeddings leveraging only information from the adjacency graph. These resulting token embeddings are evaluated when input to a CNN architecture as it has yielded the best results in most of the cases (see Sections 3.4.2, 3.4.3, 3.4.4). The results for all datasets considered so far are in Table 3.20: *SATokE 300d only-adjacency*.

One can observe that the results are consistently worse than all the other results obtained with token embeddings that take into account syntactic information as well. This empirically outlines the importance of using syntax for the creation of token embeddings and suggests that the successful results obtained by SATokE are not due to and could not be reproduced leveraging local context only. While a more in-depth analysis can be performed to look into the individual contribution of different syntactic relations and possibly a different model can be used to better capture adjacency information, the current results demonstrate that the improvement comes from the syntactic information as well and not only from adjacency.

Architecture / Embeddings	Datasets							
	"Inappropriate"				*"Appropriate"*			
	MR	CR	SUBJ	SST-2	TREC	MSRPC	SICK	SNLI-20k
LSTM								
SATokE 300d	74.39	73.00	91.19	79.60	84.40	69.27	63.59	56.76
SATokE 300d single-Punct	74.34	73.79	88.95	79.71	80.20	67.71	61.25	55.76
CNN								
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.60**	**70.32**	**62.53**	**57.72**
SATokE 300d single-Punct	75.15	76.26	89.18	80.37	88.00	69.91	60.82	55.45

Table 3.21: Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the proposed syntactically-aware token embeddings (SATokE 300d) to token embeddings computed using "collapsed" punctuation information under a single tag (SATokE 300d single-Punct). All embeddings are fixed. Best results are in **bold**.

Ablation study - impact of punctuation

Another set of experiments aims to determine whether keeping the different punctuation information within the graph of each sentence is beneficial for the end results or, on the contrary, if removing such information can hurt the results on the end task. As detailed in Section 3.3.2, punctuation embeddings are used as provided by GloVe whenever available and are learned along with the graph of the sentence otherwise. An alternative to that would be to keep the punctuation information only during the parsing phase and later collapse all punctuation signs into a single generic tag. Table 3.21 shows results of training models that consider such a setting. The results obtained when merging all punctuation signs are generally lower than those obtained in the initial setting, except for marginal increases in scores for the CR and SST-2 datasets when using an LSTM architecture. This may indicate that in general keeping a finer-grained notion of punctuation can be beneficial.

Ablation study - impact of merging relations

The original proposal maintains the different syntactic relations provided by the parser as separate matrices within each sentence's tensor. However, this implies that the infrequent relations

will be sampled less and consequently may end up having less reliable representations. To account for that and also to investigate to what extent keeping a certain level of granularity for relations is required, a set of experiments use embeddings created from merged relations.

In order to achieve this, three categories of relations are considered: those that refer to a subject (SUBJ), object (OBJ) or a modifier (MOD). For each of these, a single matrix will hold all related relations: in the case of SUBJ for example, instead of having separate matrices for the different types of SUBJ: nsubj (nominal subject), nsubjpass (nominal subject passive), csubj (clausal subject) and csubjpass (clausal subject passive), only one single relation is considered with the general SUBJ label. For the case of OBJ, the merged relations are: dobj (direct object), obj (object) and pobj (object of a preposition), while the MOD relation holds the merging of advmod (adverbial modifier), amod (adjectival modifier), nounmod (modifier of nominal), npmod (noun phrase as adverbial modifier), nummod (numeric modifier) and quantmod (modifier of quantifier) as provided by the parser.

The results of such an approach should be: smaller tensors (as the number of relations reduces), more dense matrices (since multiple modifiers can exist within the same sentence, although of different kind, all the information related to these modifiers will be stored within the same matrix in the sentence's tensor) and consequently a faster learning procedure. Table 3.22 presents results obtained using such token embeddings. The score obtained are generally lower than in the original setting, with the exception of the CR and SST-2 datasets when using an LSTM architecture. As it was the case for punctuation as well, this may indicate it may be beneficial to keep a certain granularity level for these relations. Nevertheless, there are some datasets for which the difference in scores between the two settings is not very high (at least when using an LSTM): MR, SST-2, MSRPC and SNLI-20k.

Token embeddings from the 2-step setup

As the best results using the proposed token embeddings for almost all datasets are obtained in the 1-step setup using a CNN architecture, only such an architecture is considered for testing the

Architecture / Embeddings	Datasets							
	"Inappropriate"				*"Appropriate"*			
	MR	CR	SUBJ	SST-2	TREC	MSRPC	SICK	SNLI-20k
LSTM								
SATokE 300d	74.39	73.00	91.19	79.60	84.40	69.27	63.59	56.76
SATokE 300d rels-merged	74.37	75.35	89.12	79.87	78.80	69.10	61.23	56.54
CNN								
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.60**	**70.32**	**62.53**	**57.72**
SATokE 300d rels-merged	75.21	77.08	89.29	79.98	86.20	68.92	60.88	56.64

Table 3.22: Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the proposed syntactically-aware token embeddings (SATokE 300d) to token embeddings computed using merged relations information (SATokE 300d rels-merged). All embeddings are fixed. Best results are in **bold**.

token embeddings obtained in the 2-steps setup. The results using these token embeddings are slightly lower than those obtained in the 1-step setting as it can be seen from Table 3.23. This is an expected result and the difference between the two can mark the trade-off between having a complete encoding of the sentences (when relations are learned along with the tokens and thus from the same domain) and having a faster performance (when relations are reused and may come from different domains). Thus the scores reflect a domain adaptation effect, reflecting the domain mismatch between the data used to train relations and tokens.

It is possible that using a larger training corpus to learn relations embeddings might improve on the results of inferred token embeddings. Nevertheless it can be noticed that the results obtained when using the token embeddings from the 2-step setup are still better than results obtained using standard word type embeddings (even when these are tuned) across all datasets with the CNN architecture.

Qualitative analysis

A further look can be taken at challenging examples in which the produced SATokE token embeddings perform better than the standard word type embeddings. Some examples from the

Architecture / Embeddings	Datasets							
	"Inappropriate"				*"Appropriate"*			
	MR	CR	SUBJ	SST-2	TREC	MSRPC	SICK	SNLI-20k
CNN *Standard pre-trained Word Type embeddings*								
GloVe 300d	74.92	76.65	91.07	80.15	78.80	66.89	60.23	55.84
GloVe 300d (fine-tuned)	75.31	76.20	91.08	80.15	79.00	65.25	58.11	55.48
Current proposal Token embeddings								
SATokE 300d	**76.72**	**78.81**	**91.73**	**82.13**	**92.60**	**70.32**	**62.53**	**57.72**
SATokE 300d 2-step	75.83	78.11	91.34	81.85	89.80	70.20	62.47	56.94

Table 3.23: Sentence classification and sentence pair classification results in terms of accuracy (%). Comparison of the proposed syntactically-aware token embeddings (SATokE 300d) to token embeddings computed using the 2-step setup, leveraging pre-trained relations embeddings (SATokE 300d 2-step). All embeddings are fixed. Best results are in **bold**.

S1: a dog is chasing another and is holding a stick in its mouth. S2: a dog is chasing a stick and holding another dog in its mouth.
S1: a baby is licking a dog. S2: a dog is licking a baby.
S1: a man is holding a sign and is seeking money. S2: a man is seeking a sign and is holding some money.
S1: the woman is picking up the baby kangaroo. S2: a kangaroo is picking up the woman's baby.

Table 3.24: Examples of challenging sentence pairs correctly classified using the SATokE token embeddings

SICK dataset are presented in Table 3.24. It can be noticed that the sentences on which token embeddings outperform type embeddings tend to be those which require an understanding that goes beyond word surface level: the token embeddings seem to provide a useful representation in these cases.

Finally, specific examples of sentences with polysemous words are shown in Table 3.25. For each sentence, first the polysemous word (in bold) is found and then a list of the most similar word types (by cosine similarity) to the token representation of the searched word is provided. For comparison the list of most similar word types to the type representation of the same word

Sentence	Nearest types to token	Nearest types to type
things go terribly wrong for the honest **bank** manager [...]	banks, banking, credit, central, investment, financial	banks, banking, credit, financial, investment
a tan dog is splashing in the water on the **bank** of a pond.	fed, money, branch, shore, wall	
i've read about the issues with the scroll **bar**, but they must have taken care of that now.	bars, menu, screen, shop, buttons, camera	bars, cafe, pub, lounge, room, shop, hotel
[...] first cartoon to look as if it were being shown on the unknown_nn television screen of a sports **bar**.	bars, cafe, lounge, hostess, pub	
if so, then this movie will **touch** your soul...	touches, senses, lightness, sweetness	touches, hand, screen
we are fully prepared to roll out the [**touch** - screen] machines for the 2004 presidential primary [...]	touches, buttons, handheld, screen, click, devices	

Table 3.25: Examples of most similar words to a given token

(namely its GloVe embedding) is also provided. It can be observed the token embeddings manage to capture meaning and distinguish between the senses of a word, although they have not been explicitly trained on a word sense disambiguation task. This is another clue that the produced token embeddings efficiently encode local context information obtained from the graph.

3.5 Application to discourse analysis

3.5.1 Implicit discourse relation classification

An area worth exploring the benefits of using token embeddings on is that of discourse analysis. Automatically identifying discourse relations is considered to be helpful for many downstream NLP tasks such as question answering, machine translation or automatic summarization (Dai and Huang, 2018). While identifying discourse relations in the presence of explicit connectives has proven to be relatively easy, with accuracy scores around 93% (Pitler et al., 2008), it is more challenging to identify these relations in the absence of textual cues. Nevertheless, a lot of research effort has been focused on this task along with the release of the Penn Discourse Treebank (PDTB) (Prasad et al., 2008), the largest annotated corpus of discourse relations.

In the PDTB, documents are annotated following the predicate-argument structure. More specif-

ically, a discourse connective (e.g. *so, but, because*) is treated as a predicate that takes two text spans around it as its arguments, further denoted as *Arg*-1 and *Arg*-2. The discourse connective is structurally attached to *Arg*-2 and can be either explicit or implicit. Each argument can be a single sentence, a clause or multiple sentences. In the PDTB only pairs of adjacent sentences can contain implicit relations. An example of an implicit discourse relation is provided below: the implicit connective "so" is not part of the original argument text but is added to illustrate the idea.

- ***Arg*-1:** "A lot of investor confidence comes from the fact that they can speak to us," he says.
  ***Arg*-2:** [**so**] "To maintain that dialogue is absolutely crucial."
  Class: Cause

The task of recognizing implicit discourse relations is typically approached as a classification problem, with the two arguments as input and their implicit discourse relation as the label to predict.

In implicit discourse relation classification literature, some work leverages additional information from unlabeled data by removing the existing explicit connectives (Rutherford and Xue, 2015). However, it has been proven that the nature of the natively implicit data may be different from that of an artificially constructed implicit relation corpus from data with explicit connectives (Lin et al., 2009). Other work makes use of labeled and unlabeled data (implicit and explicit) coming from different corpora in order to classify discourse relations in multi-task learning frameworks (Lan et al., 2013; Liu et al., 2016; Lan et al., 2017). In the lack of (or sometimes in addition to) exploring additional unlabeled data, the focus of most works has been either on the representation of the two arguments involved in a discourse relation or on modelling the interaction between them. This interaction has been modelled through attention mechanisms (Lan et al., 2017), through the use of features derived from word pairs (Chen et al., 2016; Lei et al., 2017) or by directly modelling the argument pair jointly (Liu et al., 2016).

Regardless of the chosen approach however, accurately representing the arguments in the discourse relation is key to building a reliable model. Although earlier works focused on the use of different feature sets as input to classification models: word pairs, part-of-speech tags, context information etc. (Pitler et al., 2009; Rutherford and Xue, 2014), little attention has been offered to varying the input features of recent deep neural network-based approaches to implicit discourse relation classification and to how these can influence the quality of the output of such models. That is, most current approaches rely on standard pre-trained word embeddings to model the arguments of a discourse relation (Qin et al., 2017; Wang et al., 2017; Dai and Huang, 2018). The reason stems from their success across a variety of tasks (Zou et al., 2013; Bansal et al., 2014; Tang et al., 2015) but also from the fact that they have been proven to perform best on implicit discourse relation classification as well (Braud and Denis, 2015).

Further, to account for pre-trained word embeddings limitations and to integrate additional knowledge, some work employs complementary features. Ji and Eisenstein (2015) note that it is difficult to fully recover the semantics of arguments using only surface level features and propose representing each argument using bottom-up compositional operations over its constituency parse tree along with a top-down approach for modelling coreferent entity mentions. Others successfully complement the use of word embeddings with extra linguistic features like part-of-speech tag embeddings or named entity tag embeddings (Dai and Huang, 2018). Qin et al. (2016a) consider character level information to enhance the word embeddings representations. An alternative is to learn distributional word representations tailored specifically for implicit discourse relation classification (Braud and Denis, 2016).

However, information from syntactic dependencies, previously proven to be beneficial for the task (Lin et al., 2009), is not integrated in any of these models. Moreover, following the intuition that word pairs and information derived from constituency parse trees matters, this has been mostly integrated in the architecture of the models through attention mechanisms rather than in the input embeddings themselves.

One of the challenges of automatically identifying implicit discourse relations is the fact that

it requires an in-depth semantic understanding of the text fragments involved in such relations. So since the task of detecting implicit discourse relations requires semantic understanding and since semantic understanding relies on encoding the word meaning in its context (Qin et al., 2016a), it is natural to investigate the use of token representations for this task.

A step can be taken in this direction by analyzing the usefulness of the proposed SATokE token embeddings for the task of implicit discourse relation classification (Popa et al., 2019a,b). The computed SATokE contextualized representations of words could be beneficial for the task as they leverage information from the sentence dependency parse to improve argument representation (which has been previously shown to be beneficial for detecting implicit discourse relations in traditional models (Lin et al., 2009)). They also have the benefit of encoding the information about the structure of the sentence from a dependency point of view in the representations themselves and not in the parameters of the end task model.

The contribution of the proposed token embeddings to the task is analyzed and compared to different standard word type representations and to the ELMo token embeddings. The proposed token representations offer improvements over traditional word representations in all considered cases. Moreover, experimental results show that the proposed representations achieve near state-of-the-art results when input to standard neural network architectures, surpassing complex models that use additional data and consider the interaction between arguments. This represents a first attempt to investigate the use of token embeddings for the task of implicit discourse relation classification and to analyze the impact of using information from syntactic dependencies as input to deep learning models for this task.

3.5.2 Data

Throughout the experiments, the Penn Discourse Treebank (PDTB) 2.0 dataset [10] (Prasad et al., 2008) is used, the largest annotated corpus of discourse relations covering 2312 Wall Street

[10] http://www.seas.upenn.edu/ pdtb/

Journal (WSJ) articles. The dataset contains a total of 16224 argument pairs annotated with their corresponding explicit and implicit discourse connectives at three levels of granularity: class, type and subtype. Level-1 contains four semantic classes: Comparison, Contingency, Expansion and Temporal, whereas level-2 contains 16 types that refine the relation senses to provide finer semantic distinctions. Level-3 is typically ignored in the literature as it provides too fine-grained subtypes and is not present for all the types (Lin et al., 2009). Some examples of such argument pairs along with their class, type and subtype annotations are further provided. All examples are taken from the training data.

1. ***Arg*-1:** They motivate sales people with commissions.

   ***Arg*-2:** Jewelry makers rarely pay commissions and aren't expected to anytime soon.

   Class: Comparison;

   Type: Comparison.Contrast;

   Subtype: Comparison.Contrast.Juxtaposition;

2. ***Arg*-1:** But the issue is stickier than it seems.

   ***Arg*-2:** France, Britain and Italy all have light tanks they would like to keep out of the talks.

   Class: Contingency;

   Type: Contingency.Cause;

   Subtype: Contingency.Cause.Reason;

3. ***Arg*-1:** But even though NATO negotiators have only 10 months left under the Bush timetable, they are still wrestling over such seemingly fundamental questions as "What is a tank".

   ***Arg*-2:** Five of the six categories of weapons under negotiation haven't even been defined.

   Class: Expansion;

   Type: Expansion.Restatement;

   Subtype: Expansion.Restatement.Specification;

4. ***Arg*-1:** Our pilot simply laughed, fired up the burner and with another blast of flame lifted us, oh, a good 12-inches above the water level.
   ***Arg*-2:** We scuttled along for a few feet before he plunged us into the drink again.
   Class: Temporal;
   Type: Temporal.Asynchronous;
   Subtype: Temporal.Asynchronous.Precedence;

In all the experiments only argument pairs annotated with *implicit* discourse relations are considered and data marked with explicit connectives is not leveraged in any way. To enable comparisons with previous work, two popular experimental setups are followed by performing multi-class classification on both level-1 and level-2. Although some work evaluates "one-versus-all" classifiers for each individual class considered, recent studies focus on the multi-class classification scenario which is more natural and realistic (Rutherford and Xue, 2014).

One split adopted is that of Lin et al. (2009), with sections 2-21 from PDTB used for training, section 22 for development and section 23 for test. This is further denoted as the PDTB-Lin split and used to perform multi-class classification for level-2 types. Similarly to Lin et al. (2009), the least frequent 5 types are removed as they account for only 9 examples in the training data and no examples in the development and test sets. The 11 types considered are: Expansion.List, Expansion.Conjunction, Expansion.Instantiation, Expansion.Restatement, Expansion.Alternative, Comparison.Concession, Comparison.Contrast, Contingency.Cause, Contingency.Pragmatic cause, Temporal.Asynchronous and Temporal.Synchrony. In PDTB about 2.2% of the implicit relations are annotated with two types. Following previous work, during training, instances with more than one annotation are considered as multiple instances, each with one type annotation. During testing, a correct prediction is one that matches one of the annotated types. The number of instances in each split and their class distribution are shown in Table 3.26.

A second split considered is that of Pitler et al. (2009). Here sections 2-20, 0-1 and 21-22 from PDTB are used as training, development and test sets, respectively. This split is denoted as

Class	Train	Dev	Test	Total
Comparison.Concession	195	5	5	205
Comparison.Contrast	1653	88	127	1868
Contingency.Cause	3423	123	200	3746
Contingency.Pragmatic cause	68	2	5	75
Expansion.Conjunction	2968	117	118	3203
Expansion.Instantiation	1176	48	72	1296
Expansion.Restatement	2569	104	190	2863
Expansion.Alternative	159	2	15	176
Expansion.List	345	5	30	380
Temporal.Asynchronous	582	29	13	624
Temporal.Synchrony	213	19	5	237
Total	13351	542	780	14673

Table 3.26: Data statistics for the PDTB-Lin split

Class	Train	Dev	Test	Total
Comparison	1944	152	197	2293
Contingency	3346	279	292	3917
Expansion	7011	574	671	8256
Temporal	760	85	64	909
Total	13061	1090	1224	15375

Table 3.27: Data statistics for the PDTB-Pitler split

PDTB-Pitler and used to report results for level-1 multi-class classification to enable comparisons to previous work. Additionally, it should be noted that focusing on the level-1 relations enables one to be theory-neutral as these represent the four core discourse relations that various discourse analytic theories seem to converge on (Wang et al., 2012). The number of instances for each set and their class distribution are shown in Table 3.27.

Data Appropriateness

A similar analysis to that from Section 3.4.1 can be done for the implicit discourse relation classification data. Table 3.28 outlines some characteristics of this data: average sentence length and percentage of sentences for which a SUBJ and/or an OBJ relationship has not been detected by the parser. According to the separation done in Section 3.4.1, the discourse datasets may not qualify as *"appropriate"* due to the high average sentence length (19.42) which could result

	PDTB
average sentence length	19.24
%sent without SUBJ rel	4.04%
%sent without OBJ rel	8.22%

Table 3.28: Data characteristics for the PDTB data.

.

in parsing challenges and further propagate their effects on the token embeddings computation. However, similarly to the MSRPC dataset, it is important to note that the discourse data seems to have very few sentence without a subject or object relationship, 4.04% and 8.22% respectively, which might indicate generally well-formed sentences, appropriate for evaluation in the current setup.

3.5.3 Experimental setup

Many neural network models and variants have been proposed to learn the semantic representation of each argument in a discourse pair. However, most of them rely on the same basic neural network building blocks: LSTM and/or CNN. In the following, the proposed SATokE token embeddings are evaluated as input to these neural network models, as part of a standard architecture as outlined in Figure 3.7. Additionally a gated mechanism is exploited, that enables controlling the flow of information and minimally modelling the interaction between the two arguments in a discourse relation (Qin et al., 2016b).

No interaction between the arguments First the vector representations of each argument are obtained either through an LSTM or a CNN encoding on top of word type embeddings or the proposed token embeddings. Then the vectors corresponding to the arguments are concatenated into a vector of the pair $\boldsymbol{v} = [\boldsymbol{h}_n, \boldsymbol{h}_m]$ for an LSTM encoding and $\boldsymbol{v} = [\boldsymbol{z}_1, \boldsymbol{z}_2]$ for a CNN encoding. The representation of the pair is further passed to a fully connected layer, followed

by a softmax layer to obtain the probability distribution over labels. This approach, however, does not focus on modelling the interaction between the two arguments in the discourse relation, an important aspect when predicting discourse relations, as pointed out by previous work (Chen et al., 2016; Lan et al., 2017).

Collaborative Gated Neural Network The Collaborative Gated Neural Network (CGNN) architecture was introduced by Qin et al. (2016b) for the task of implicit discourse relation classification and represents a model providing minimal interaction between the arguments. In the proposed model, the arguments are individually modelled using convolutional neural networks that share parameters of the convolution and pooling operations among each other. Then an additional gated unit is used for feature transformation.

The input to the CGNN unit is the vector representing the concatenation of the arguments $\boldsymbol{v} = [\boldsymbol{z}_1, \boldsymbol{z}_2]$. Then the set of transformations are given by:

$$\hat{c} = \tanh(\boldsymbol{W}^c \cdot \boldsymbol{v} + \boldsymbol{b}^c)$$

$$\boldsymbol{g}_i = \sigma(\boldsymbol{W}^i \cdot \boldsymbol{v} + \boldsymbol{b}^i)$$

$$\boldsymbol{g}_o = \sigma(\boldsymbol{W}^o \cdot \boldsymbol{v} + \boldsymbol{b}^o)$$

$$\boldsymbol{c} = \hat{c} \odot \boldsymbol{g}_i$$

$$\boldsymbol{h} = \tanh(\boldsymbol{c}) \odot \boldsymbol{g}_o$$

where $\odot$ denotes the element-wise multiplication, σ denotes the sigmoid function, $\hat{c}$ and $\boldsymbol{c}$ are inner cells, $\boldsymbol{g}_i$ and $\boldsymbol{g}_o$ are the two gated operations and $\boldsymbol{W}^i$, $\boldsymbol{W}^o$ and $\boldsymbol{W}^c$ are parameters of the model.

The output of the CGNN unit is the transformed vector $\boldsymbol{h}$ which is further passed to a softmax layer. Finally, the training objective is defined as the cross-entropy loss between the output of the softmax layer and the class labels. The architecture can be seen in Figure 3.16. Note that

the vector of the pair $\boldsymbol{v}$ is denoted by $[\boldsymbol{Enc}_{Arg\text{-}1}, \boldsymbol{Enc}_{Arg\text{-}2}]$ in Figure 3.16.

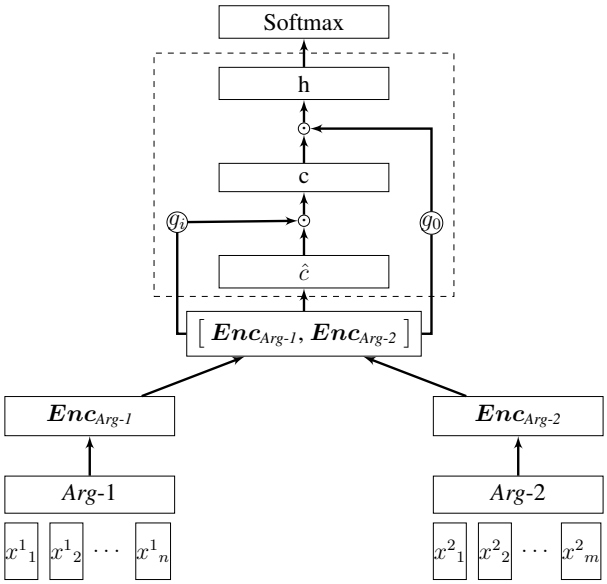


Figure 3.16: CGNN (Qin et al., 2016b)

3.5.4 Implementation details

For the computation of token embeddings the best results are obtained when using an initial learning rate of 10^{-3}, a sampling factor of 5x and the margin parameter γ set to 10 for the PDTB-Pitler split. For the PDTB-Lin split, the best results are obtained when using an initial learning rate of 10^{-4}, a negative sampling factor of 1x and the margin parameter γ set to 1.

A distribution of the dependency relations considered for the task is presented in Figure 3.17. The most frequent relations present in the dataset are: prep (preposition), pobj (object of preposition), det (determiner), compound and nsubj (nominal subject). About one third of all relations have frequencies above the average, while the "unknown_rel" relation represents a very low proportion of the data.

According to the information in Table 3.29 almost half of the sentences in the PDTB data have more than one subject relation detected, indicating the presence of composed structures. This, combined with a high average sentence length as outlined in Table 3.28 and the variance present

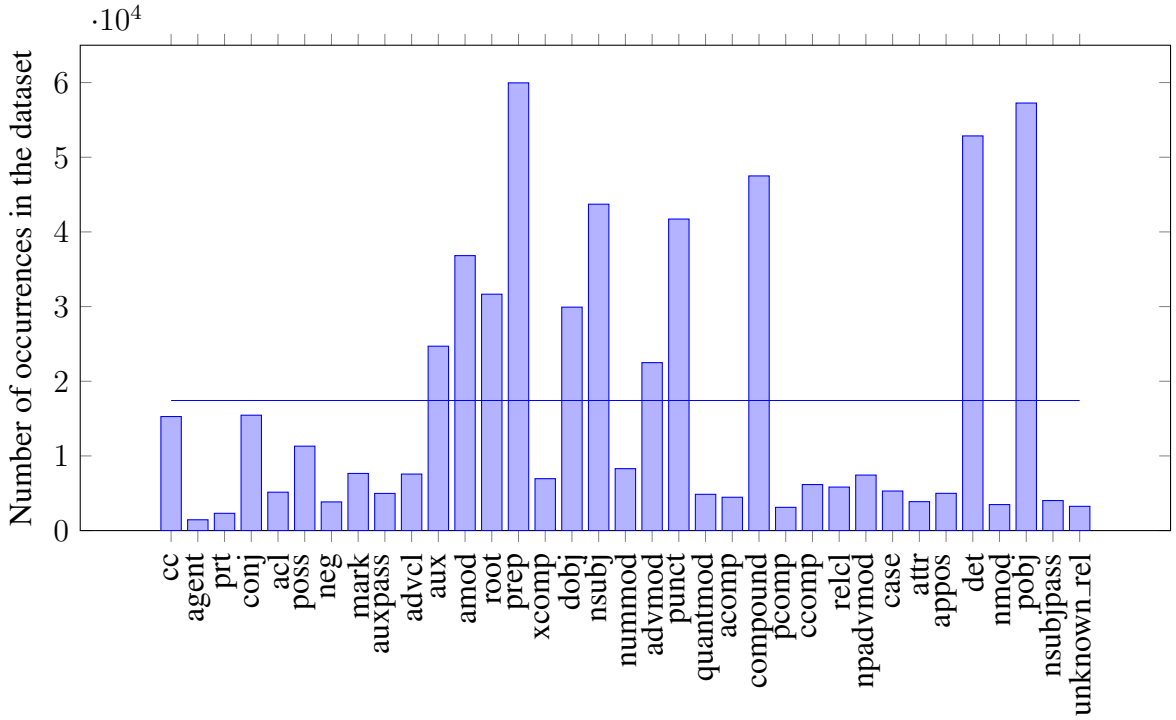


Figure 3.17: Distribution of the considered syntactic relations in the PDTB data.

in the frequency of syntactic relations can indicate a certain complexity of the data, making the PDTB one of the challenging datasets for the proposed token embeddings evaluation.

For the optimisation of the implicit discourse relation classification task Adam (Kingma and Ba, 2014) is used and the parameters of the models are tuned on the development set. The parameters that lead to the reported results for each split are shown in Table 3.30, with α_{LR} denoting the initial learning rate, dp the dropout probability, nbh the size of the fully connected layer, szb the size of the batches and nbf the number of filters. For the CNN and GCNN architectures the filters used are of sizes 3, 4 and 5. The parameters of the CNN are shared across the two arguments.

In the following a comparison is provided between the proposed token embeddings and standard pre-trained word type embeddings. This is followed by a parallel between the reported results and state-of-the-art systems aimed at solving the task as well as a comparison to the ELMo token embeddings. All word type embeddings considered are 300-dimensional. The SATokE token embeddings proposed in the current work are 300-dimensional as well, while the ELMo

	PDTB
avg subj / sent	1.56
% sent / ds w/ $>$ 1 subj rel	41.26%
% sent / ds_subj w/ $>$ 1 subj rel	43.00%
avg link distance	2.66

Table 3.29: Elements of data complexity (from top to bottom): average number of subject relationships per sentence, percentage of sentences out of the whole dataset that have more than 1 subject relationship, percentage of sentences that have more than 1 subject relationship out of the sentences that have at least 1 subject relationship, average dependency link distance.

Data split	LSTM runs			CNN runs					CGNN runs				
	α_{LR}	dp	nbh	α_{LR}	dp	nbf	szb	nbh	α_{LR}	dp	nbf	szb	nbh
PDTB-Lin	10^{-4}	0.7	150	10^{-5}	0.8	600	16	300	10^{-4}	0.7	128	64	768
PDTB-Pitler	10^{-5}	0.7	512	10^{-5}	0.6	600	16	512	10^{-4}	0.6	600	64	3600

Table 3.30: Parameters used for the reported results on each data split

token embeddings are 1024-dimensional. The considered ELMo representations are obtained as a weighted sum of the different layers of the deep bidirectional language model, with the weights being trainable. This is supported by the claim of Peters et al. (2018) that the semantic and syntactic aspects are captured in different intermediate layers and including representations from all layers improves overall performance. In that sense, combining the intermediate layers for ELMo represents a trade-off between the two aspects. In the case of SATokE, the semantic and syntactic aspects are weighted equally given that the ratio between the two loses is set to $\alpha = 0.5$. Finally a series of tests are conducted over model variations in order to analyze the impact of the syntactic information and to justify its usefulness.

3.5.5 Comparison to word type embeddings

Table 3.31 presents the results of running three sets of experiments on the PDTB-Lin data using the different encoding schemes: the two LSTM and CNN encoding approaches presented in 3.3.1 and the collaborative architecture (CGNN) presented of Qin et al. (2016b). A compari-

son is provided between the results obtained using different input features: standard word type embeddings often employed in literature and pre-trained using local context over large corpora (Pennington et al., 2014; Mikolov et al., 2013b), word type embeddings pre-trained using dependency contexts (Levy and Goldberg, 2014) and the proposed SATokE token representations.

	LSTM	CNN	CGNN
GloVe (Pennington et al., 2014)	38.97	38.25	39.03
Word2Vec (Mikolov et al., 2013b)	36.92	37.33	37.07
Deps-WE (Levy and Goldberg, 2014)	36.00	34.98	34.98
SATokE	**40.51**	**42.55**	**43.08**

Table 3.31: Results for level-2 multi-class classification on the PDTB-Lin split in terms of accuracy (%). Comparison to word type embeddings. Best results are in **bold**.

It can be observed that using the proposed token embeddings as input yields important improvements over all the word type embeddings and across all the architectures considered. The obtained improvements are between 1.5% and 4.5% when using an LSTM architecture, between 3.7% and 7.5% with a CNN encoder and between 4% and 8% with the CGNN architecture.

Although the CGNN architecture represents a model of minimal interaction, it is still more complex than the LSTM and CNN encoders. Consequently, in 2 out of 4 cases (using GloVe embeddings and the proposed token embeddings) the CGNN architecture reaches the highest accuracy scores. One would expect that a more advanced interaction model would bring even further improvements.

Unsurprinsingly, feeding the proposed token embeddings as input to an LSTM encoder only improves the results by up to 4.5% over the word type embeddings. Since the proposed SATokE token embeddings already encode positional information, passing them through an LSTM network is less beneficial than for word embeddings. This improvement is lower than the improvement of up to 8% observed when using the CGNN architecture over the same embeddings.

On the contrary, for the embeddings trained on dependency contexts (Levy and Goldberg, 2014) the best scores are obtained when using the LSTM architecture. This is not a surprising result as those embeddings may lack information about the immediate local context, since they were

constructed using only dependency contexts. This is therefore complemented by the use of an LSTM encoder that considers local context. For the pre-trained Word2Vec embeddings, the choice of architecture does not influence much the results.

Another observation is that using as input features the dependency-based word embeddings yields consistently worse results than all the other embeddings considered. This result is consistent with the analysis in Ghannay et al. (2016) and with the analysis in 3.4.5. Again, contrary to a first interpretation, this does not imply that dependency information is not useful for semantic tasks: it can be seen that the induced SATokE token representations based strongly on dependency information outperform all the other representations. Therefore token embeddings computed using dependency information can bring higher value than generic word type embeddings trained on such contexts. The observed results support this belief with improvements between 4.5% and 8% in absolute accuracy value between the word embeddings trained on dependency contexts and the proposed token embeddings that use dependency information.

3.5.6 Comparison to state-of-the-art systems

A comparison to the related work for the fine-grained level-2 multi-class classification of the PDTB-Lin split is further provided. The results can be seen in Table 3.32. Although the scores obtained using the SATokE token embeddings are close to those of state-of-the-art systems, it is important to note that they either use additional information from non-implicit relations to train their models or consider more complex architectures:

- Lin et al. (2009) exploit and analyze the use of features derived from constituent and dependency parse trees as well as word pair features as input to a traditional maximum entropy classifier;
- Qin et al. (2016a) use a hybrid architecture in which they enhance the word embeddings representations with character level information derived from stacked convolutional and bidirectional LSTM layers;

Model	Accuracy (%)
Lin et al. (2009)	40.20
Qin et al. (2016a)	43.81
Qin et al. (2017)	44.65
CNN + SATokE	42.55
CGNN + SATokE	43.08

Table 3.32: Results for level-2 multi-class classification on the PDTB-Lin split in terms of accuracy (%). Comparison to related work.

- Qin et al. (2017) use a more complex model in an adversarial framework leveraging explicit discourse connectives. Their adversarial setup is composed of a network fed only data with implicit relations, a network fed data augmented with connectives, a discriminator aiming at distinguishing between the features produced by the two and a final classifier to determine the discourse relation between the two arguments.

To enable further comparisons to other models in the literature, a set of experiments is performed on the level-1 multi-class classification using the PDTB-Pitler split. Although this split is less fine-grained that the PDTB-Lin one, a wider variety of works have considered it. Table 3.33 compares the results using the CNN and CGNN architectures with the proposed SATokE token embeddings to the results reported in the literature for this split. Additional details are provided regarding the resources used by related work, the approaches employed and the number of parameters required. Estimations for parameter count are provided whenever the details in the corresponding work are not sufficient to compute an exact number. All numbers assume an input embedding size of 300.

While the results obtained are competitive to related work, the systems we compare to differ from the current proposal from several points of view. Some work uses additional data from different corpora or containing explicit connectives (Rutherford and Xue, 2015; Liu et al., 2016; Ji et al., 2016; Lan et al., 2017; Dai and Huang, 2018). Other work focuses on complex architectures to model the interaction between the two arguments (Liu et al., 2016; Liu and Li, 2016; Ji et al., 2016; Lan et al., 2017; Dai and Huang, 2018). Further details about each work are provided in the following:

- Rutherford and Xue (2015) collect additional training data through distant supervision over freely omissible explicit discourse connectives;

- Liu et al. (2016) use a multi-task neural network framework leveraging a combination of different discourse corpora. They use CNNs to represent the pairs of arguments, creating both unique and shared representations across the different tasks and consider additional surface-level features.

- Lan et al. (2017) use an attention-based LSTM to model the interaction between the two arguments integrated in a multi-task joint learning setting. They also leverage explicit discourse relations and unlabelled external data.

- Liu and Li (2016) use a multi-level attention mechanism to repeatedly read the arguments involved in a discourse relation along with an external short-term memory to keep track of the exploited information. The arguments are modeled using bidirectional LSTM networks.

- Dai and Huang (2018) model inter-dependencies between arguments and the sequence patterns of their discourse connectives by positioning them in the wider context of paragraphs.

- Ji et al. (2016) model jointly the discourse relation and the arguments with a latent variable RNN-based architecture using additional data from non-implicit relations. Their proposed probabilistic model is used to infer discourse relations given the representations of the arguments as well as to do discourse-aware language modelling.

- Wang et al. (2017) focus on the representation of the arguments: they propose Tree-LSTM and Tree-GRU models to encode the arguments. They use the structure of the constituency parse trees to recursively compose semantics bottom-up and information from the constituent tags in order to control for semantic composition and induce grammatical information.[11]

Model	Accuracy (%)	Additional data	Arg Interaction	Multi-task	$\|\theta\|$
Wang et al. (2017)	56.04 / *59.85*	-	-	-	$6.7M$
Rutherford and Xue (2015)	*57.10*	✓	-	-	NA
Liu et al. (2016)	*57.27*	✓	✓	✓	$> 1M$
Lan et al. (2017)	*57.39*	✓	✓	✓	$> 1M$
Liu and Li (2016)	*57.57*	-	✓	-	$6.7M$
Dai and Huang (2018)	*58.20*	✓	✓	-	$> 4M$
Ji et al. (2016)	*59.50*	✓	✓	-	$5.5M$
LSTM + GloVe 300d	54.60	-	-	-	$1.5M$
CNN + GloVe 300d	56.42	-	-	-	$4M$
LSTM + ELMo 1024d	56.97	-	-	-	$1.5M$
CNN + ELMo 1024d	57.81	-	-	-	$4M$
LSTM + SATokE 300d	56.63	-	-	-	$1.5M$
CNN + SATokE 300d	**58.83**	-	-	-	$4M$
CGNN + SATokE 300d	57.90	-	✓	-	$41M$

Table 3.33: Results for level-1 multi-class classification on the PDTB-Pitler split in terms of accuracy (%). Comparison of the proposed SATokE token embeddings to related work, to GloVe word type embeddings and to ELMo token embeddings. In *italic* and above the first double line are reported scores in the literature. All non-italic scores are (re)produced in the current work.

It can be observed that by using a simple CNN encoder with the SATokE token embeddings as input features, one can obtain near state-of-the-art results, surpassing most complex models which exploit external data or model the interaction between arguments or their positioning in a wider context. The best scores obtained without additional data in a non multi-task setting and without modelling argument interaction are those resulting from the CNN setup with SATokE token embeddings as input. This setup also represents the one with the lowest number of parameters for the given performance level. Additionally, the scores obtained in this setup also surpass results obtained by the any of the architectures when provided the GloVe word type embeddings or the ELMo token embeddings as input.

As in the case of the PDTB-Lin split, among the 3 architectures considered, using an LSTM over the token embeddings yields the lowest results. This may indicate that modelling the arguments using a simple LSTM may not be sufficient to encode the required information for the task. Another possible explanation can be related to the fact that each argument in the

[11]The reproduction of the scores reported by Wang et al. (2017) was impossible, despite re-creating the setup described in the paper with the code provided by the authors.

Model	Accuracy (%)
Wang et al. (2017) + GloVe 50d	55.70
Wang et al. (2017) + GloVe 300d	56.04
Wang et al. (2017) + SATokE 300d	56.55
CNN + SATokE 300d	**58.83**
CGNN + SATokE 300d	57.90

Table 3.34: Results for level-1 multi-class classification on the PDTB-Pitler split in terms of accuracy (%). Comparison to originally reported results and re-created settings of Wang et al. (2017).

pair is represented only by the last hidden state of the LSTM. Thus subsequent layers only have access to that information, while the CNN architecture has access to the computed token embeddings at each position. Alternatively it can indicate that the SATokE token embeddings are not fully exploitable when input to an LSTM either because the token embeddings already encode positional information or because their syntactic component is less adapted for use in a sequential manner.

The CNN+SATokE setup also surpasses the CGNN+SATokE run, suggesting that even though the CGNN architecture models the interaction between arguments, it may not constitute a powerful enough architecture for this setup: as noted previously it is indeed a minimal interaction model. Modelling the interactions between the arguments is orthogonal to how one chooses to model the arguments - which is the focus of the current work. Thus the two can bring complementary benefits in predicting the discourse relation holding between two arguments.

Although Lin et al. (2009) suggested the importance of including dependency-related information when modelling the arguments of a discourse relation, none of the recent models investigates the use of syntactic dependencies for this task. Moreover, contrary to the current proposal, none of them can encode arbitrary graphs. It is also important to note that, as observed from Table 3.33, the number of parameters required by the state-of-the-art models (Ji et al., 2016; Wang et al., 2017) is higher than the number of parameters used by the CNN encoder along with SATokE representations.

3.5.7 Extension: combining dependency information with constituent parse features

Finally, inspired by the findings in Lin et al. (2009) regarding the benefit of combining information from syntactic dependencies with constituent parse features, one additional set of experiments is run using the architecture presented in Wang et al. (2017). This architecture is particularly fit for the current goal as it leverages constituency-based parse trees. Table 3.34 shows all results. Unfortunately, there are issues reproducing the results reported by Wang et al. (2017), despite re-creating the setup described in the paper with the code provided by the authors. The re-run of their method with the reported parameters and 50-dimensional GloVe embeddings yielded only 55.7% accuracy, a drop of slightly over 4% from the originally reported results.

To investigate to what extent token embeddings can be beneficial as input to the proposed architecture in Wang et al. (2017), two additional experiments were considered: one leveraging 300-dimensional GloVe embeddings (for fair comparison) and one considering the proposed 300-dimensional SATokE token embeddings. It can be observed that the token embeddings yield a slight improvement over using word type embeddings, confirming once again the benefit of having contextualized representations. However, the results obtained in both cases are approximately 2% lower than the results obtained using the proposed token embeddings in the CNN and CGNN architectures. This indicates that further investigation might be required into how to efficiently combine the dependency information with knowledge derived from constituency-based parse trees for this task.

3.5.8 Ablation studies - impact of syntax - model variations

While modelling the words in their context seems to be beneficial for the task, one can further investigate to what extent syntax plays an important role in the obtained scores. An additional set of experiments analyze the impact of using dependency information in the computation

of the token embeddings. All results are reported in Table 3.35. The parameters for the token embeddings computation are set to the ones that obtained the best results in the default scenario, hereafter denoted *SATokE*[12]. Then two comparative settings are considered: token embeddings computed without the adjacency relation *SATokE-adjacency*, and without all syntactic relations *SATokE-syntax* respectively. The *SATokE-syntax* refers to a setting in which information about individual syntactic relations is removed, yet one matrix is kept to account for any link between tokens as derived from the syntactic dependency tree (unlabeled). The decrease of performance in results shows that both adjacency relation and syntax play an important role in the final result. However, the results seem to degrade slightly more when information about syntax is removed from the token computation than when adjacency information is not present.

Further experiments take into account only certain types of dependency relations considered to be highly relevant. The *SATokEAdj+SUBJ+OBJ+MOD+1rel* setup refers to tokens computed taking into account information derived from the SUBJ, OBJ and MOD dependency relations along with adjacency information. In this setting, one additional relation stands for all the other dependency relations that occur between the tokens with frequency higher than the previously set threshold. This additional relation is modelled by one single matrix in the tensor. Consequently, this reduces the sparsity as it aggregates information that would otherwise be at the basis of several matrices. The resulting tensors have thus less relations, also making the learning process faster. It can be observed that the scores obtained in these settings are not far from the best obtained scores overall. This may indicate that selecting only certain dependency relations to use and merging all the rest, could provide good results, while at the same time reducing the computation time and fighting the data sparsity issue when creating the token embeddings.

The two settings *fine-grained* and *coarse-grained* denote whether the original distinctions have been maintained or not for the more granular dependency relations. For example in the *fine-grained* setting, the nsubj (nominal subject), nsubjpass (nominal subject passive), csubj (clausal subject) and csubjpass (clausal subject passive) relations are all kept separate and thus constitute

[12]The tokens used for the reported results in Table 3.33.

Model variations	Accuracy (%)
Original model	
SATokE	**58.83**
Removing information	
SATokE-adjacency	57.81
SATokE-syntax	57.65
Limited information	
SATokEAdj+SUBJ+OBJ+MOD+1rel_fine-grained	58.07
SATokEAdj+SUBJ+OBJ+MOD+1rel_coarse-grained	58.57
Filtering	
SATokE-SUBJ	58.83
SATokE-SUBJ-OBJ	57.90
SATokE-SUBJ-OBJ-MOD	57.56

Table 3.35: Results for level-1 multi-class classification on the PDTB-Pitler split using variations of the proposed token embeddings model with a CNN architecture. Results are reported in terms of accuracy (%). Best results are in **bold**.

separate matrices in the tensor of a sentence, while in the *coarse-grained* setting they are all merged into the SUBJ relation and modelled using a single matrix in the tensor. It can be observed from Table 3.35 that using the *coarse-grained* setup yields a 0.5% increase over its corresponding *fine-grained* counterpart. This can be related to the sparsity issue: as Figure 3.17 shows, there are not enough examples of lower granularity dependency relations in the corpus to account for having a separate matrix for each of these relations and to drive efficient learning: see the low frequencies of nmod (modifier of nominal), nsubjpass (nominal subject passive) and quantmod (modifier of quantifier). Thus having a single general relation that stands for multiple fine-grained relations could be beneficial in the lack of high amounts of data. Overall, these results also suggest that there may be settings in which other dependency relations could be merged to obtain better results.

Finally, another set of experiments is considered in which certain dependency relations considered important are iteratively filtered out from the token embeddings computation. We thus obtain the *SATokE-SUBJ* setting which corresponds to tokens computed without information coming from the SUBJ dependency relations, *SATokE-SUBJ-OBJ* to denote tokens computed without information regarding the SUBJ and OBJ dependency relations and *SATokE-SUBJ-*

OBJ-MOD corresponding to tokens computed without information coming from the SUBJ, OBJ and MOD dependency relations. It can be observed that, for the most part, results degrade the more information is removed from the computation of the tokens.

3.6 Conclusion

Acknowledging the need for a contextualized representation of words, existing approaches in current literature leverage machine translation (McCann et al., 2017) or language modelling (Peters et al., 2017, 2018) on large amounts of data to enable the construction of word token embeddings. However, none of the currently existing methods leverages directly the sentence structure from a dependency point of view to create these token representations. Moreover, most of the token embeddings are further used in combination with word type embeddings (pre-trained on large amounts of data), character-level embeddings or other feature vectors and are then provided as input features for different classifiers to solve various natural language understanding tasks. Strictly determining the contribution of token embeddings obtained by methods available in the literature is thus biased by their typical evaluation in combination with different other features.

In contrast to existing work, the current proposal provides an unsupervised method to obtain token embeddings by jointly encoding information from the sentence dependency parse tree and local context information (adjacency). This information is injected directly into the token representations, such that each token would reflect the whole sentence structure from its perspective. Additionally, the proposed method provides embeddings for the relations that hold between tokens, which can be further leveraged to compute token embeddings on new corpora. In the proposed model, sentences are modelled as graphs, with tokens standing for the nodes and relations between them denoting the edges in the graphs. Embeddings for tokens and relations are then obtained by factorizing these graphs into corresponding structures. A ranking objective is used in order to reconstruct the relations between tokens in the graph of each sentence as

detailed in Section 3.2.

The proposed representations have been evaluated on various natural language understanding tasks ranging from sentiment analysis to paraphrase detection, textual entailment recognition and implicit discourse relation classification in Section 3.4 and Section 3.5 also outlined in Popa et al. (2019a,b). A discussion was provided in Section 3.4.1 with respect to the *"appropriateness"* of considering such datasets for evaluation based on the characteristics of the sentences contained in them: given the proposed token embeddings method relies heavily on parsing information, it is important to ensure the consistency of such information when providing it as input to the token embeddings computation. The obtained results show indeed that datasets having certain characteristics (and being adapted to syntactic analysis) can benefit more from the use of the proposed syntactically-aware token embeddings.

Several discussions have been provided as well around other characteristics of the datasets such as the complexity of the structures present in sentences and the frequency distribution of dependency relations and how these may impact performance on the end task. However, it is important to note that the proposed SATokE token embeddings obtain competitive results on challenging datasets as it can be seen from the experiments in Section 3.4 and Section 3.5. For example they outperform all word type embeddings considered as well as the ELMo token embeddings on both implicit discourse relation classification (PDTB dataset) and paraphrase detection (MSRPC dataset), both datasets being among the ones with the highest average sentence length, the highest average dependency link distance and rather important variations among the frequencies of dependency relations, which made the embeddings computation process more challenging.

When provided as input features to commonly-employed neural network architectures, the proposed SATokE token embeddings have shown clear superiority to general purpose word type embeddings. This has been observed including when the word type embeddings were pretrained on large corpora using dependency information or were enhanced with positional information or self-attention. Additionally, in some cases, the SATokE token embeddings fed

to simple neural network-based models provide comparable or even increased performance to more complex state-of-the-art methods that use word type embeddings as input, while at the same time requiring less parameters as shown in Section 3.5.

Comparisons to other token embeddings methods yielded similar results with the type of data being evaluated on and the size of the compared token embeddings playing an important role in the final scores. Analysis conducted in Section 3.4 outlined multiple aspects: firstly, considering ELMo token embeddings of comparable dimensionality (256d) to that of SATokE (300d) incurs a performance drop. This suggests that at least a part of the difference in performance between ELMo (1024d) and SATokE (300d) may be attributed to dimensionality.

Further, datasets involving sentences with rich vocabulary, but not necessarily challenging syntactic structures favor the token embeddings trained on large amounts of data with a language modelling objective. However, when dealing with sentences for which the understanding requires going beyond word level and that pose a structural challenge, using the proposed token embeddings brings benefits over alternative token embeddings methods. In that sense one possible solution would be to use different weights for the semantic and syntactic aspects of the loss. This could boost performance on datasets where the syntactic aspect is less present (for example reviews with sentences sometimes lacking verbs or the SUBJ relation). Alternatively, or in addition to this, it would be interesting to use the proposed method to train relation embeddings on large amounts of data and further leverage them for the computation of token embeddings on any new given dataset. Furthermore, one could analyze to what extent giving different weights to different syntactic relations can influence the scores.

Further, the amount of unknown words in a dataset has been shown to directly correlate to the drop in performance with respect to the related work token embeddings methods. This can be explained by the fact that related work estimates embeddings for unknown words based on their character-level representation, while this is not integrated in the current proposal. Adding character-level information to the proposed token embeddings could however constitute an interesting experiment for the future.

Finally, qualitative analysis in Section 3.4.5 has shown that the proposed token embeddings distinguish between the different senses of words although they have not been explicitly trained to do so. Further analysis can be conducted in this sense by checking whether similarities do exist between tokens that play the same functional role in different sentences (for example fulfilling the subject role in a sentence).

In general, when constructing the token embeddings with the proposed method, using syntactic information yields improvements in scores over just using adjacency information. This has been observed across all datasets as detailed in Sections 3.4.5 and 3.5. One interesting future direction though, can also be that of using relations from a semantic parser within the same framework. Additionally, one could also consider enriching the graph of a sentence with information from the part-of-speech tags of the words in the sentence or any other linguistic knowledge. From this point of view, the method proposed here is expandable and can deal with arbitrary graphs.

As a conclusion, having a contextualized word representation is beneficial for natural language understanding tasks. The current proposal contributes to the space of token embeddings computation by providing an unsupervised method to encode linguistic information and sentence structure into token representations directly. The obtained token embeddings provide competitive results on the tasks considered in this work (Popa et al., 2019a,b).

Chapter 4

Conclusion

Most approaches to natural language processing rely on vectorial word representations as their input signal. These representations can be either pre-trained on large corpora and used as the initialization point for subsequent tasks or learned along with the task at hand. While some work focused on developing different methods for creating word representations that can be used successfully across different tasks (Mikolov et al., 2013b; Levy and Goldberg, 2014), other work specialized these representations into satisfying certain linguistic constraints that pertained to individual tasks (Vulic and Mrksic, 2018).

In that sense, Chapter 2 proposes a framework for inducing an entailment vector space in which words are represented by vectors denoting probabilities of features being known, thus enabling the modelling of information inclusion. A set of proposed entailment operators are tested as part of hyponymy detection evaluation and shown to obtain competitive results on a widely used benchmark dataset. The proposal and the obtained results are also reflected in Henderson and Popa (2016).

However, a challenging aspect is going beyond single word representations and encoding longer units of text. With many existing architectures in the literature, yet no general consensus over the one that encodes the best compositionality notions, combining the representations of words

to form representations of sentences remains an open area of research.

At the same time, encoding words within the context they appear in can enrich their representations and could eventually help in enabling compositionality. One possible way to contextualize word representations is through a model that considers both local and syntactic contexts for any given word in a sentence. Further, such a model must condition the representation of each word on the representations of all the other words it is related to in that sentence, such as to be able to encode the structure of the sentence within the representations of each word. Some compositional aspects could be therefore injected within the word representations themselves as in the proposal described in Chapter 3 and in Popa et al. (2019a,b). Analysis of the obtained contextualized word representations across various sentence understanding tasks outline the benefits of using such an approach and widen the areas of exploration for context aware word representations.

4.1 Summary of Thesis Achievements

The following outline the main achievements presented in the current thesis:

- a framework for modelling entailment within a vector space, relying on information inclusion, contributing to the area of specialized vector spaces (Henderson and Popa, 2016);
- a set of operators that can be used to reason about entailment within the vector space;
- an evaluation of the proposed entailment framework and operators for the task of hyponymy detection as an instance of lexical entailment recognition;
- a flexible framework for explicitly incorporating linguistic knowledge into contextualized word representations (Popa et al., 2019a,b);
- a method for constructing syntax-aware contextualized word token representations (SATokE) that implicitly include aspects of the structure of the sentence they are part of (Popa et al.,

2019a,b);

- a comparison of the proposed token representations to other contextualized word representations proposed in the literature;
- an evaluation of the constructed token representations on a wide range of sentence understanding tasks;
- a set of ablation studies over the proposed model as well as a discussion of the obtained results.

4.2 Publications

Peer - reviewed International Conferences Articles

- Henderson, James and Popa, Diana Nicoleta. (2016). A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL.*
- Popa, Diana Nicoleta, Perez, Julien, Henderson, James, and Gaussier, Eric. (2019). Implicit discourse relation classification with syntax-aware contextualized word representations. In *Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference, FLAIRS-32*.

Peer - reviewed International Journal Articles

- Popa, Diana Nicoleta, Perez, Julien, Henderson, James, and Gaussier, Eric. (2019). Towards Syntax-aware Token Embeddings. In *Journal of Natural Language Engineering, JNLE* (under review).

Peer - reviewed National Conferences Articles

- Popa, Diana Nicoleta, Perez, Julien, Henderson, James, and Gaussier, Eric. (2019). SATokE: How can Syntax-Aware Contextualized Word Representations Benefit Implicit Discourse Relation Classification? In *Conférence sur l'Apprentissage automatique, CAp*.

4.3 Future Work

While the proposal in Chapter 2 encodes lexical entailment within the vector space, more work can be done to explore the potential of methods to extract information about entailment from large corpora. Alternatively, in a similar fashion, one could imagine encoding other types of relations between words instead of (or in addition to) enforcing the entailment constraints within a vector space. In that sense a possible future direction concerns investigating further methods to inject additional linguistic constraints into the vectorial representations themselves as well as investigating to what extent doing so could benefit different natural language understanding tasks.

Also, one possible future direction concerns merging the two proposals from Chapter 2 and Chapter 3 into a model that goes beyond lexical entailment to reason about textual entailment. One could thus imagine enforcing the information inclusion constraints along with ensuring contextualized representations within the same model. This can be achieved either by reusing the lexical entailment framework proposed in Chapter 2 on top of the architecture in Chapter 3 or by deriving a new architecture to consider both entailment and contextuality all-together.

Concerning the proposal in Chapter 3, more effort can be put into investigating how different types of syntactic relations can affect the performance of the created representations on different tasks. This could further lead to a model in which different relations are weighted differently to obtain the token embeddings. Also only some relations could be taken into account, while others can be completely discarded. Similarly, the syntactic and the semantic aspects can be

weighted differently in the computation of the token embeddings. This aspect can be related to investigating whether it is the syntactic or the semantic aspects that play a more important role depending on the task. Relations from a semantic parser can also be used instead of, or in addition to, the syntactic dependencies.

Since the proposal in Chapter 3 is suitable for encoding arbitrary graphs, much other information can be included in the model, like for example that coming from external knowledge resources. It is also possible to consider including further linguistic constraints into the proposed model, such as for example information concerning the part-of-speech tags or information from the constituency parse tree. This could be achieved either through regularization terms (like the one proposed in the current approach to ensure the link to pre-trained word embeddings) or through the use of an attributes matrix holding the desired linguistic properties that one may want to enforce. Since previous work investigated ways to jointly factorize tensors and matrices (Singh et al., 2015), such a properties matrix could be factorized along with the tensor holding the graph of the sentence.

Also other factorization methods can be considered along with different options for the structures used to encode the words and the relations between them, like for example considering vectors instead of matrices for relations. Whether such a decrease in the representation space is indeed beneficial or even sufficient, to still represent the interactions between words within the parse tree, is in itself a question. Varying the nature of the pre-trained word embeddings that serve as constraint for the semantic nature of the token embeddings is also an option. Additionally, one could consider adding a component that leverages character-level information or an alternative to modelling the semantic aspects in the representations of rare words. Investigating such an approach could help assess the benefits of the proposed representations by improving on the comparison to methods that do handle these aspects.

From an evaluation point of view, one could consider more complex architectures on top of the proposed token embeddings. Thus it would be possible to investigate to what extent other architectures are able to make use of the information present in the contextualized word repre-

sentations and whether a certain type of architecture is more suitable than others. Additionally, different tasks could be considered, such as those that rely on more that just word similarity or relatedness and that require an in-depth sentence understanding.

One particularly interesting direction concerns investigating to what extent the proposed representations pass the challenging compositionality-awareness tests proposed in recent literature (Dasgupta et al., 2018; Ettinger et al., 2018) and if they do not, in which way token representations can be improved to do so. In the same space, it would be very interesting to test the token embeddings as part of adversarial setups such as that proposed by Nie et al. (2019). In that sense one could expect positive results, given the performance of the proposed token embeddings on the SICK dataset [1] as seen from the evaluation in Section 3.4.4.

Further, more work can be done to investigate alternative methods to encode contextual information in the vector representations within parametric settings. One potential approach consists in creating contextualized word representations by reusing a neural network encoder architecture trained for dependency parsing, similarly to how related work reuses encoders trained for language modelling or machine translation.

[1] It it worth noting that 7% of the sentence pairs in the SICK dataset are formed by sentences having the same vocabulary but different meanings due to the way these words are combined with each-other. In that sense, evaluating only that proportion of the SICK dataset could be seen as an instance of some of the tests proposed by Dasgupta et al. (2018).

Bibliography

Bach, E. (1989). *Informal lectures on formal semantics*. State University of New York Press.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*.

Bansal, M., Gimpel, K., and Livescu, K. (2014). Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Baroni, M., Bernardi, R., Do, N.-Q., and Shan, C.-c. (2012). Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

Baroni, M. and Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Journal of Computational Linguistics*.

Baroni, M. and Lenci, A. (2011). How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*.

Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Bentivogli, L., Bernardi, R., Marelli, M., Menini, S., Baroni, M., and Zamparelli, R. (2016). SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Journal of Language Resources and Evaluation*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Braud, C. and Denis, P. (2015). Comparing word representations for implicit discourse relation classification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Braud, C. and Denis, P. (2016). Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. (2018). e-SNLI: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems 31*.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. (2018). Universal sentence encoder for english. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Chang, H.-S., Wang, Z., Vilnis, L., and McCallum, A. (2018). Distributional inclusion vector embedding for unsupervised hypernymy detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Chen, J., Zhang, Q., Liu, P., Qiu, X., and Huang, X. (2016). Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of 54th Annual Meeting of Association for Computational Linguistics*.

Chen, Q., Zhu, X., Ling, Z.-H., Wei, S., Jiang, H., and Inkpen, D. (2017). Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chen, X., Liu, Z., and Sun, M. (2014). A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Cheng, J. and Kartsaklis, D. (2015). Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Clark, S., Coecke, B., and Sadrzadeh, M. (2008). A compositional distributional model of meaning. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

Clark, S. D. and Pulman, S. G. (2007). Combining symbolic and distributional models of meaning. In *AAAI Spring Symposium: Quantum Interaction*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing:

Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Conneau, A. and Kiela, D. (2018). SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Dagan, I., Glickman, O., and Magnini, B. (2006). The PASCAL recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*.

Dagan, I., Roth, D., Sammons, M., and Zanzotto, F. M. (2013). Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*.

Dai, Z. and Huang, R. (2018). Improving implicit discourse relation classification by modeling inter-dependencies of discourse units in a paragraph. *ArXiv e-prints*.

Dasgupta, I., Guo, D., Stuhlmüller, A., Gershman, S. J., and Goodman, N. D. (2018). Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Dasigi, P., Ammar, W., Dyer, C., and Hovy, E. H. (2017). Ontology-aware token embeddings for prepositional phrase attachment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dhingra, B., Shallue, C. J., Norouzi, M., Dai, A. M., and Dahl, G. E. (2018). Embedding text in hyperbolic spaces. In *TextGraphs@Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ding, Z., Xia, R., Yu, J., Li, X., and Yang, J. (2018). Densely connected bidirectional LSTM with applications to sentence classification. In *Proceedings of the International Conference on Natural Language Processing and Chinese Computing*.

Dolan, B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Erk, K. and Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ettinger, A., Elgohary, A., Phillips, C., and Resnik, P. (2018). Assessing composition in sentence vector representations. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Foltz, P. W., Kintsch, W., and Landauer, T. K. (1998). The measurement of textual coherence with latent semantic analysis. *Discourse Processes*.

Frege, G. (1884). *Die Grundlagen der Arithmetik: eine logisch mathematische Untersuchung über den Begriff der Zahl*. W. Koebner.

Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. (2015). Learning semantic hierarchies: A continuous vector space approach. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. (2013). PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., and Xu, W. (2015). Are you talking to a machine? Dataset and methods for multilingual image question answering. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*.

Geffet, M. and Dagan, I. (2005). The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*.

Ghannay, S., Favre, B., Estve, Y., and Camelin, N. (2016). Word embeddings evaluation and combination. In *Journal of Language Resources and Evaluation*.

Gidaris, S. and Komodakis, N. (2015). Object detection via a multi-region and semantic segmentation-aware CNN model. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Journal of Computational Linguistics*.

Glavaš, G. and Vulić, I. (2018). Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks.*

Grefenstette, E. and Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Guo, J., Che, W., Wang, H., and Liu, T. (2014). Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*.

Harris, Z. (1954). Distributional structure. *Word*.

He, H., Gimpel, K., and Lin, J. (2015). Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Henderson, J. and Popa, D. N. (2016). A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Hill, F., Reichart, R., and Korhonen, A. (2015). SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Journal of Computational Linguistics*.

Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Journal of Computational Linguistics*.

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

Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*.

Irsoy, O. and Cardie, C. (2013). Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*.

Issa, F., Damonte, M., Cohen, S. B., Yan, X., and Chang, Y. (2018). Abstract meaning representation for paraphrase detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. (2015). Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Ji, Y. and Eisenstein, J. (2015). One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*.

Ji, Y., Haffari, G., and Eisenstein, J. (2016). A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kartsaklis, D. and Sadrzadeh, M. (2013). Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Kawakami, K. and Dyer, C. (2015). Learning to represent words in context with multilingual supervision. *CoRR*, abs/1511.04623.

Kenter, T. and de Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*.

Kim, S., Hong, J.-H., Kang, I., and Kwak, N. (2019). Semantic sentence matching with densely-connected recurrent and co-attentive information. *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the 2014 International Conference on Learning Representations*.

Kintsch, W. (2001). Predication. *Cognitive science*.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., and Fidler, S. (2015). Skip-thought vectors. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*.

Kokkinos, F. and Potamianos, A. (2017). Structural attention neural networks for improved sentiment analysis. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Komninos, A. and Manandhar, S. (2016). Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-geffet, M. (2010). Directional distributional similarity for lexical inference. *Journal of Natural Language Engineering*.

Kruszewski, G., Paperno, D., and Baroni, M. (2015). Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*.

Labeau, M. and Allauzen, A. (2017). Character and subword-based word representation for neural language modeling prediction.

Lan, M., Wang, J., Wu, Y., Niu, Z.-Y., and Wang, H. (2017). Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Lan, M., Xu, Y., and Niu, Z. (2013). Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Lan, Y. and Jiang, J. (2018). Embedding WordNet knowledge for textual entailment. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Landauer, T., Laham, D., and Rehder, R. (1997). How well can passage meaning be derived without using word order? A comparison of latent semantic analysis and humans. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society.*

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In

Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32.

Lei, W., Wang, X., Liu, M., Ilievski, I., He, X., and Kan, M.-Y. (2017). SWIM: a simple word interaction model for implicit discourse relation recognition. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

Lenci, A. and Benotto, G. (2012). Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*.

Levy, O., Dagan, I., and Goldberger, J. (2014). Focused entailment graphs for open IE propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Levy, O. and Goldberg, Y. (2014). Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Levy, O., Remus, S., Biemann, C., and Dagan, I. (2015). Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Li, X. and Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*.

Lin, Z., Kan, M.-Y., and Ng, H. T. (2009). Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ling, W., Luís, T., Marujo, L., Astudillo, R. F., Amir, S., Dyer, C., Black, A. W., and Trancoso, I. (2015). Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Liu, P., Qiu, X., and Huang, X. (2015). Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

Liu, Y. and Li, S. (2016). Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of Empirical Methods in Natural Language Processing*.

Liu, Y., Li, S., Zhang, X., and Sui, Z. (2016). Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

MacAvaney, S. and Zeldes, A. (2018). A deeper look into dependency-based word embeddings. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The Penn Treebank. In *Journal of Computational Linguistics - Special issue on using large corpora*.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. (2017). Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30*.

McCarthy, D. and Carroll, J. (2003). Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Journal of Computational Linguistics*.

Melamud, O., Goldberger, J., and Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.

Milajevs, D., Kartsaklis, D., Sadrzadeh, M., and Purver, M. (2014). Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Miller, G. A. (1995). WordNet: A lexical database for english. *Communications of the ACM*.

Mirkin, S., Specia, L., Cancedda, N., Dagan, I., Dymetman, M., and Szpektor, I. (2009). Source-language entailment modeling for translating unknown terms. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*.

Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. In *Journal of Cognitive Science*.

Mrkšic, N., OSéaghdha, D., Thomson, B., Gašic, M., Rojas-Barahona, L., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mrkšić, N., Vulić, I., Ó Séaghdha, D., Leviant, I., Reichart, R., Gašić, M., Korhonen, A., and Young, S. (2017). Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*.

Necsulescu, S., Mendes, S., Jurgens, D., Bel, N., and Navigli, R. (2015). Reading between

the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Nguyen, K. A., Köper, M., Schulte im Walde, S., and Vu, N. T. (2017). Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems 30*.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*. ACM.

Nie, Y., Wang, Y., and Bansal, M. (2019). Analyzing compositionality-sensitivity of NLI models. *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence*.

Padó, S., Galley, M., Jurafsky, D., and Manning, C. D. (2009). Textual entailment features for machine translation evaluation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*.

Padó, S. and Lapata, M. (2007). Dependency-based construction of semantic space models. *Journal of Computational Linguistics*.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Partee, B. (1995). Lexical semantics and compositionality. *An invitation to cognitive science: Language*.

Partee, B., Ter Meulen, A., and E Wall, R. (1990). *Mathematical Methods in Linguistics*.

Pasunuru, R., Guo, H., and Bansal, M. (2017). Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Peters, M., Ammar, W., Bhagavatula, C., and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Pitler, E., Louis, A., and Nenkova, A. (2009). Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Pitler, E., Raghupathy, M., Mehta, H., Nenkova, A., Lee, A., and Joshi, A. (2008). Easily

identifiable discourse relations. *Proceedings of the 22nd International Conference on Computational Linguistics*.

Plate, T. (1991). Holographic reduced representations: Convolution algebra for compositional distributed representations. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*.

Popa, D. N., Perez, J., Henderson, J., and Gaussier, E. (2019a). Implicit discourse relation classification with syntax-aware contextualized word representations. In *Proceedings of the 32nd International Florida Artificial Intelligence Research Society Conference*.

Popa, D. N., Perez, J., Henderson, J., and Gaussier, E. (2019b). Satoke: How can syntax-aware contextualized word representations benefit implicit discourse relation classification? *Conférence sur l'Apprentissage automatique, CAp*.

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B. (2008). The Penn Discourse TreeBank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

Qin, L., Zhang, Z., and Zhao, H. (2016a). Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*.

Qin, L., Zhang, Z., and Zhao, H. (2016b). A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of Empirical Methods in Natural Language Processing*.

Qin, L., Zhang, Z., Zhao, H., Hu, Z., and Xing, E. (2017). Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of Association for Computational Linguistics*.

Rei, M. and Briscoe, T. (2014). Looking for hyponyms in vector space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kocisky, T., and Blunsom, P. (2016). Reasoning about entailment with neural attention. In *Proceedings of the 2016 International Conference on Learning Representations*.

Roller, S., Erk, K., and Boleda, G. (2014). Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*.

Rudolph, S. and Giesbrecht, E. (2010). Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Rutherford, A. and Xue, N. (2014). Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Rutherford, A. and Xue, N. (2015). Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sacaleanu, B., Orasan, C., Spurk, C., Ou, S., Ferrandez, O., Kouylekov, M., and Negri, M. (2008). Entailment-based question answering for structured data. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*.

Salant, S. and Berant, J. (2018). Contextualized word representations for reading comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Santus, E., Lenci, A., Lu, Q., and im Walde, S. S. (2014). Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Schütze, H. (1993). Word space. In *Advances in Neural Information Processing Systems 5*.

Shelhamer, E., Long, J., and Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Shwartz, V., Santus, E., and Schlechtweg, D. (2017). Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

Singh, S., Rocktäschel, T., and Riedel, S. (2015). Towards Combined Matrix and Tensor Factorization for Universal Schema Relation Extraction. In *The North American Chapter of the Association for Computational Linguistics Workshop on Vector Space Modeling for NLP (VSM)*.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Journal of Artificial Intelligence*.

Snow, R., Jurafsky, D., and Ng, A. Y. (2006). 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*.

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., and Manning, C. D. (2011a). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*.

Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011b). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013a). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013b). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Tang, J., Qu, M., and Mei, Q. (2015). PTE: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. (2016). Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*.

Tu, L., Gimpel, K., and Livescu, K. (2017). Learning to embed words in context for syntactic tasks. *CoRR*, abs/1706.02807.

Tuan, L. A., Tay, Y., Hui, S. C., and Ng, S. K. (2016). Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Turney, P. D. and Mohammad, S. M. (2014). Experiments with three approaches to recognizing lexical entailment. *Journal of Natural Language Engineering*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30*.

Vilnis, L. and McCallum, A. (2015). Word representations via Gaussian embedding. In *Proceedings of the 2015 International Conference on Learning Representations*.

Vulic, I. and Mrksic, N. (2018). Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Vylomova, E., Rimell, L., Cohn, T., and Baldwin, T. (2015). Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *arXiv preprint arXiv:1509.01692*.

Wang, X., Li, S., Li, J., and Li, W. (2012). Implicit discourse relation recognition by selecting typical training examples. *Proceedings of the 24th International Conference on Computational Linguistics*.

Wang, Y., Li, S., Yang, J., Sun, X., and Wang, H. (2017). Tag-enhanced tree-structured neural networks for implicit discourse relation classification. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*.

Weeds, J., Clarke, D., Reffin, J., Weir, D., and Keller, B. (2014). Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*.

Weeds, J. and Weir, D. (2003). A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*.

Weeds, J., Weir, D., and McCarthy, D. (2004). Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Weir, D., Weeds, J., Reffin, J., and Kober, T. (2016). Aligning packed dependency trees: A theory of composition for distributional semantics. *Journal of Computational Linguistics*.

West, R. F. and Stanovich, K. E. (1986). Robust effects of syntactic structure on visual word processing. *Memory & Cognition*.

Westera, M. and Boleda, G. (2019). Don't blame distributional semantics it can't do entailment. In *Proceedings of the 13th International Conference on Computational Semantics*.

Widdows, D. (2008). Semantic vector products: Some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

Wittgenstein, L. (1953). *Philosophical investigations*. Blackwell.

Xu, C., Bai, Y., Bian, J., Gao, B., , Liu, X., and Liu, T.-Y. (2014). RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*.

Yao Zhou, C. L. and Pan, Y. (2016). Modelling sentence pairs with tree-structured attentive encoder. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Yessenalina, A. and Cardie, C. (2011). Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yu, M. and Dredze, M. (2014). Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Yu, Z., Wang, H., Lin, X., and Wang, M. (2015). Learning term embeddings for hypernymy identification. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., and Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. (2013). Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.