Universal semantic tagging methods and their applications

by

Joan Ginés i Ametllé

Jointly submitted to

Saarland University

in partial fulfilment of the requirements for the

Master's Degree in Language Science and Technology

and to

University of Groningen

in partial fulfilment of the requirements for the

Master's Degree in Linguistics

Supervisors:

Prof. Dr. Yusuke Miyao
Digital Content and Media Sciences Research Division
National Institute of Informatics

Prof. Dr. Johan Bos
Center for Language and Cognition
University of Groningen

Prof. Dr. Dietrich Klakow
Department of Language Science and Technology
Saarland University

Erasmus Mundus LCT – 31st August 2018

Declaration of authorship

I hereby confirm that the thesis here presented is my own and original work, with all references and forms of assistance explicitly acknowledged.

Eidesstattliche erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Verklaring eigen werk

Hierbij verklaar ik, dat het werk dat in deze scriptie gepresenteerd wordt origineel is en dat ik geen gebruik heb gemaakt van andere bronnen dan die welke in de tekst worden genoemd.

Signed: Joan Ginés i Ametllé

Date: 31/08/2018

A l'Erina, la meva família i els meus amics,

i també a tots aquells que sempre em donen suport encara que gran distància ens allunyi.

Abstract

Universal semantic tagging is the recently proposed task of assigning certain semantic categories (semantic tags or sem-tags) to each token in a given text fragment. These semantic tags are designed to individually represent the meaning contribution of each token, and are independent to the characteristics of any particular language. Because of their special properties, semantic tags provide various advantages over Part-Of-Speech (POS) tags when considering problems that involve semantic analysis, and are suitable for cross-lingual semantic parsing. At the time of writing, the most recent version of the Universal Semantic Tagset consists of 73 fine-grained semantic tags grouped into 13 meta-tags or coarse-grained semantic tags.

Possibly due to its short-lived existence, there are not many established results for universal semantic tagging, and its usefulness has not yet been quantitatively measured. Thus, in the first part of this thesis, we are concerned with formulating neural models for universal semantic tagging building on recent research results for tagging tasks and the annotated data provided by the Parallel Meaning Bank (PMB). The customizable software tool implemented to do so is able to obtain a performance on the task superior to other approaches and constitutes a solid baseline for future research.

In the second part of this thesis, we explore the potential of semantic tags for Recognizing Textual Entailment (RTE), which is the problem of discovering whether an entailment relationship exists or not between a given text and a given hypothesis in natural language. To that effect, we employ existing parsing and inference systems and replace all lexical information and POS tags with semantic tags. These modifications achieve state-of-the-art performance on the FraCas dataset, and pave the way for using the same approach on similar more sizable datasets.

Acknowledgements

Several people from different institutions have contributed to this work with either guidance, comments or indirect forms of support. Nonetheless, I wish to first extend my gratitude to my fellow Erasmus Mundus classmates for the time spent together through this long and at times difficult journey.

I am also grateful to the students and staff of the National Institute of Informatics. This includes Prof. Dr. Yusuke Miyao for his weekly advice, Namgi Han and Juan Ignacio Navarro Horñiacek for our interesting discussions, Takahiro Kondo and Akira Miyazawa for keeping our research group together, and many other kind and very capable people.

A very special mention goes out to Dr. Pascual Martínez Gómez and Goran Topić from the National Institute of Advanced Industrial Science and Technology for their technical suggestions at various levels.

I would also like to thank the staff at Saarland University, in particular Prof. Dr. Dietrich Klakow for his help as a supervisor, and Bobbye Pernice and Dr. Jürgen Trouvain for their throughout assistance whenever I encountered any difficulties.

Finally, last but by no means least, I would like to mention the staff at the University of Groningen, and thank Dr. Gosse Bouma for his support with all formal procedures and Prof. Dr. Johan Bos for bringing me closer to the field of semantics.

In the event that Prof. Dr. Alexander Koller, Dr. Noortje Venhuizen or Prof. Dr. Michael Biehl are reading this work, I would just like to tell them that they are fantastic teachers.

Contents

List of abbreviations

ANN	Artificial Neural Network
BGRU	Bidirectional Gated Recurrent Unit
BLSTM	Bidirectional Long Short-Term Memory
CCG	Combinatory Categorial Grammar
CRF	Conditional Random Field
EM	Expectation Maximization
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MFC	Most Frequent Class
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MWE	Multi-Word Expression
NE	Named Entity
OOV	Out-Of-Vocabulary
PCCG	Probabilistic Combinatory Categorial Grammar
PMB	Parallel Meaning Bank
POS	Part-Of-Speech
ReLU	Rectified Linear Unit
ResNet	Residual Network
RNN	Recurrent Neural Network
RTE	Recognizing Textual Entailment
TBL	Transformation-Based Learning
WSJ	Wall-Street Journal

1. Introduction

As opposed to other problems within the field of natural language processing such as information extraction or summarization, semantic parsing seeks to obtain complete and formal meaning representations of natural language sentences. Many well-performing approaches to obtaining these representations rely on the principle of compositional semantics, which states that the meaning representation of a phrase is determined by appropriately combining the meaning representations of its constituents [BCS+04, MMMB].

Nonetheless, finding the appropriate lexical semantics to assign to individual tokens is a problem that goes far beyond the surface form of words or the syntactic purpose they serve. For example, one can consider the token *have* in the following Part-Of-Speech (POS) tagged sentences extracted from the Parallel Meaning Bank (PMB) [ABE+17], only to arrive at the conclusion that the 3 occurrences have clearly distinct semantic contributions:

$$\texttt{I}^{\boxed{\texttt{PRP}}}\ \texttt{have}^{\boxed{\texttt{VBP}}}\ \texttt{an}^{\boxed{\texttt{DT}}}\ \texttt{orange}^{\boxed{\texttt{NN}}}\ \texttt{and}^{\boxed{\texttt{CC}}}\ \texttt{an}^{\boxed{\texttt{DT}}}\ \texttt{apple}^{\boxed{\texttt{NN}}}\ \texttt{.}^{\boxed{\texttt{.}}} \tag{1.1}$$

$$\texttt{We}^{\boxed{\texttt{PRP}}}\ \texttt{have}^{\boxed{\texttt{VBP}}}\ \texttt{consumed}^{\boxed{\texttt{VBN}}}\ \texttt{all}^{\boxed{\texttt{PDT}}}\ \texttt{the}^{\boxed{\texttt{DT}}}\ \texttt{natural\textasciitilde resources}^{\boxed{\texttt{NNS}}}\ \texttt{.}^{\boxed{\texttt{.}}} \tag{1.2}$$

$$\texttt{I}^{\boxed{\texttt{PRP}}}\ \texttt{have}^{\boxed{\texttt{VBP}}}\ \texttt{to}^{\boxed{\texttt{TO}}}\ \texttt{repair}^{\boxed{\texttt{VB}}}\ \texttt{the}^{\boxed{\texttt{DT}}}\ \texttt{refrigerator}^{\boxed{\texttt{NN}}}\ \texttt{.}^{\boxed{\texttt{.}}} \tag{1.3}$$

In sentence (1.1), the token *have* describes an event of possession, while denoting a perfect aspect in sentence (1.2) and being part of a modal expression in sentence (1.3). In contrast, the POS tag $\boxed{\texttt{VBP}}$ is invariably assigned to all occurrences of *have*, indicating the presence of a verb not in the third person singular form. The information provided by POS tags can indeed assist in syntactically parsing each sentence, but a more precise source of knowledge is needed for mapping word tokens to their individual lexical semantics.

Universal semantic tagging [AB17] is a recently proposed tagging task that attempts to address the shortcomings of POS tags in semantic parsing. Each associated semantic tag (or sem-tag) exclusively captures information related to the semantic contribution of each token and is agnostic to less relevant aspects such as its grammatical function. The design is such that telling the different meanings of *have* apart becomes straight-forward under semantic tags:

$$\texttt{I}^{\boxed{\texttt{PRO}}}\ \texttt{have}^{\boxed{\texttt{ENS}}}\ \texttt{an}^{\boxed{\texttt{DIS}}}\ \texttt{orange}^{\boxed{\texttt{CON}}}\ \texttt{and}^{\boxed{\texttt{GRP}}}\ \texttt{an}^{\boxed{\texttt{DIS}}}\ \texttt{apple}^{\boxed{\texttt{CON}}}\ \texttt{.}^{\boxed{\texttt{NIL}}} \tag{1.4}$$

$$\texttt{We}^{\boxed{\texttt{PRO}}}\ \texttt{have}^{\boxed{\texttt{NOW}}}\ \texttt{consumed}^{\boxed{\texttt{EXT}}}\ \texttt{all}^{\boxed{\texttt{AND}}}\ \texttt{the}^{\boxed{\texttt{DEF}}}\ \texttt{natural\textasciitilde resources}^{\boxed{\texttt{CON}}}\ \texttt{.}^{\boxed{\texttt{NIL}}} \tag{1.5}$$

$$\texttt{I}^{\boxed{\texttt{PRO}}}\ \texttt{have}^{\boxed{\texttt{NEC}}}\ \texttt{to}^{\boxed{\texttt{NIL}}}\ \texttt{repair}^{\boxed{\texttt{EXS}}}\ \texttt{the}^{\boxed{\texttt{DEF}}}\ \texttt{refrigerator}^{\boxed{\texttt{CON}}}\ \texttt{.}^{\boxed{\texttt{NIL}}} \tag{1.6}$$

Using this novel paradigm, it is possible to assign the sem-tag `ENS` to the token *have* in sentence (1.4) in order to denote an event occurring in the present. The sem-tag `NOW` can also be employed to identify the present tense contribution within the perfect aspect of the token *have* in sentence (1.5). Finally, the token *have* in sentence (1.6) can be associated to the sem-tag `NEC` to indicate necessity. The advantage of semantic tags is that tokens mapped to the same sem-tag with the same syntactic function also have the same lexical semantics, and one can manually define or otherwise learn this latter correspondence.

However, no matter how feasible it is to assign lexical semantics to word tokens according to their associated sem-tags, a model that assigns the correct sem-tags to tokens in a given text is still necessary. The present thesis proposes an implementation for such a semantic tagger using the latest available research results and resources. In particular, the aim is to conduct both an intrinsic and an extrinsic evaluation on the problem of universal semantic tagging, defining the following 2 goals:

1. Engineer an optimal semantic tagger and analyze its performance.
2. Employ the proposed tagger in a way such that it can improve the performance of an existing system for recognizing textual implications.

We begin by providing a more detailed formulation of tagging tasks in Chapter 2, including POS tagging and universal semantic tagging. Chapter 3 contains an overview of multiple approaches to tagging from traditional long-standing statistical models to modern neural network architectures, with Chapter 4 describing our proposed neural semantic tagger and its implementation. The following Chapter 5 is concerned with current methods for compositional semantic parsing, while it also addresses the Recognizing Textual Entailment (RTE) problem and how we employ the information provided by semantic tags in a specific RTE task. Future research directions and overall conclusions from this work are given in the closing Chapter 6.

2. Tagging tasks

It is commonplace for natural language processing methods to adopt a sequential approach in the form of a pipeline where the first steps attempt to extract information from basic language units. In contrast, steps occurring later in the pipeline are concerned with a higher-level linguistic analysis, attempting to capture the meaning of a given phrase or its purpose [ID10]. Since the particular family of tagging tasks simply assigns an informative label to each word token in a sequence, such tasks are performed as a prerequisite for subsequent tasks more often than not.

The labels assigned to tokens are commonly known as tags, and their possible instances differ according to their intended purpose. Let us briefly consider the problem of syntactic parsing, which attempts to discover the underlying syntactic structure of a given phrase. Because human language is inherently ambiguous, a token such as *back* can perform several roles, acting either as an adverb, a noun, an adjective, or a verb, as illustrated respectively by the following example sentences:

```
Tom put the book back on the shelf .                          (2.1)
The victim was clearly stabbed in the back .                  (2.2)
Anna entered by the back door .                               (2.3)
She knows that the goverment will back her .                  (2.4)
```

In cases like these, Part-Of-Speech (POS) tags provide valuable information by indicating the lexical category of each associated token, which enables us to make assumptions about the underlying syntactic structure [Man11]. Furthermore, POS tags are also employed in other domains such as information retrieval or even speech synthesis [JM09].

Now considering the problem of semantic parsing, the nature of the meaning contribution of each word token also suffers from ambiguity in the sense that we would like to differentiate two given tokens semantically even when they both belong to the same lexical category. Nonetheless, as we have already proved in Chapter 1, POS tags do not suffice as a source of information that enables this distinction, and thus one might want to employ a completely different set of semantic tags instead [AB17].

2.1. Part-Of-Speech tagging

The argument that knowing the lexical category of a word is helpful when trying to determine the syntactic structure of its surroundings is indeed a compelling one. For example, nouns are usually preceded by determiners or adjectives, and tend to form part of a larger noun phrase [JM09]. However, we must ask ourselves how is it possible to determine a set of fine-grained POS tags that covers all different morphological and syntactic behaviors which words can have within a sentence.

Without committing to any particular language, the linguistic community agrees that 3 major universal POS categories exist: nouns, verbs and adjectives. From there, various secondary POS categories that present similar morphological or distributional properties can be added. This being said, committing to a particular tagset is an issue subject to the target language and multiple linguistic considerations beyond the scope of this work. In the case of English, the Penn Treebank tagset [MMS93] is the most widely used [JM09].

The state-of-the-art POS tagging accuracy is over 97% for many languages, a score that prompted many researchers to claim that POS tagging is essentially a solved problem [Man11]. Nonetheless, we must mention that it is possible to obtain high tagging accuracies using relatively simple methods, and that increasing still low sentence accuracy scores has the potential to bring improvements in downstream tasks [ID10].

2.1.1. The Penn Treebank Part-Of-Speech Tagset

Most language processing applications for English employ the Penn Treebank POS tagset [MMS93] shown in Table 2.1, which has been used to label a wide range of corpora [JM09]. As we discuss in Chapter 3, having access to tagged corpora is a necessary resource for many classes of tagging models, particularly the ones that try to model statistical distributions. The most notorious corpus for English is perhaps the Brown corpus [FK79], which contains over a million words extracted from approximately 500 texts published in 1961. The Wall-Street Journal (WSJ) corpus is also a notorious resource that contains a million words from articles published in the Wall Street Journal in 1989 [JM09].

Despite the moderate number of different tags in the Penn Treebank POS tagset, it is possible to broadly classify them into closed class and open class tags. Closed class tags represent a finite set of typically function words, such as determiners (DT) or prepositions, (IN) that prominently serve a grammatical purpose and contain relatively little lexical information. Open class tags, such as singular proper nouns (NNP), represent sets can accept an arbitrary number of potentially recently coined words.

CC coordinating conj.: *and, but, or*	PRP$ possessive pronoun: *your, one's*
CD cardinal number: *one, two*	RB adverb: *quickly, never*
DT determiner: *a, the*	RBR adverb, comparative: *faster*
EX existential there: *there*	RBS adverb, superlative: *fastest*
FW foreign word: *mea culpa*	RP particle: *up, off*
IN preposition & sub. conj.: *of, in, by*	SYM symbol: *+, %, &*
JJ adjective: *yellow*	TO "to": *to*
JJR comparative adjective: *bigger*	UH interjection: *ah, oops*
JJS superlative adjective: *wildest*	VB verb base form: *eat*
LS list item marker: *1, 2, One*	VBD verb past tense: *ate*
MD modal: *can, should*	VBG verb gerund: *eating*
NN singular noun or mass: *llama*	VBN verb past participle: *eaten*
NNS plural noun: *llamas*	VBP verb non-3sg present: *eat*
NNP singular proper noun: *IBM*	VBZ verb 3sg present: *eats*
NNPS plural proper noun: *Carolinas*	WDT wh-determiner: *which, that*
PDT predeterminer: *all, both*	WP wh-pronoun: *what, who*
POS possessive ending: *'s*	WP$ wh-possessive: *whose*
PRP personal pronoun: *I, you, he*	WRB wh-adverb: *how, where*

Table 2.1: Penn Treebank POS tags excluding punctuation related tags [MMS93], with example words extracted from [JM09].

2.2. Universal semantic tagging

Semantic parsing is the task of obtaining a meaning representation of a given phrase. In this thesis, we assume that this analysis is performed by composing formal meaning representations of lexical items [BCS$^+$04] driven by syntactic derivations of Combinatory Categorial Grammars (CCG) [Ste96]. We will also adopt λ-expressions as a means of representing semantics and simply typed λ-calculus as our computational model of choice [MMMB]. These topics are covered in depth in Chaper 5, but the main relevant topic at this point is finding a way of assigning lexical semantics to tokens at a certain position within a syntactic derivation tree.

One might decide to use the lexical information provided by POS tags in order to associate lexical semantics to tokens. Such an example is shown in Equation 2.5, where we consider the phrases *glass window* and *clear window*. The token *window* is assigned to syntactic category N, denoting a nominal constituent. Similarly, both the tokens *glass* and *clear* correspond to syntactic category N/N, indicating that they combine with a nominal constituent of category N on its right to produce a larger constituent of category N.

$$\begin{array}{c} \dfrac{\text{glass}^{\boxed{\texttt{NN}}} \;/\; \text{clear}^{\boxed{\texttt{JJ}}}}{N/N} \\ \boxed{\texttt{NN}} \;\mapsto \lambda E \lambda F \lambda x.\, E(x) \wedge F(x) \\ \boxed{\texttt{JJ}} \;\mapsto \lambda E \lambda F \lambda x.\, E(x) \wedge F(x) \end{array} \quad , \quad \begin{array}{c} \dfrac{\text{window}^{\boxed{\texttt{NN}}}}{N} \\ \boxed{\texttt{NN}} \;\mapsto \lambda E \lambda x.\, E(x) \\ \end{array} \tag{2.5}$$

In the example shown in Equation 2.5, we were able to specify a set of rules that assign lexical semantics to tokens based on their syntactic categories and POS tags. Thus, tokens with syntactic category N and POS tag $\boxed{\texttt{NN}}$ will be mapped to the expression $\lambda E \lambda x.\, E(x)$. Likewise, tokens with syntactic category N/N will be mapped to the expression $\lambda E \lambda F \lambda x.\, E(x) \wedge F(x)$ when their associated POS tags are either $\boxed{\texttt{NN}}$ or $\boxed{\texttt{JJ}}$. Note that, in the leaf nodes of the syntactic derivation tree, we adopt the convention that the lexical form of the corresponding token becomes the first argument of its semantics.

However, there are many situations were POS tags do not provide sufficient information for assigning lexical semantics. The class of determiners constitutes an illustrative example [AB17], as shown in Equation 2.6 by considering the phrases *any restaurant* and *some restaurant*. Despite the fact that the tokens *any* and *some* must be associated to different semantics, POS tagging labels them both as determiners ($\boxed{\texttt{DT}}$). One must then decide on the semantics of the token associated to a syntactic category NP/N by examining its surface form. While feasible, such an approach is tedious, increases the complexity of semantic parsing and fails to generalize across different languages

$$\begin{array}{c} \dfrac{\text{any}^{\boxed{\texttt{DT}}} \;/\; \text{some}^{\boxed{\texttt{DT}}}}{NP/N} \\ \textit{'any'} \;\mapsto \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \forall x (F_1(x) \rightarrow (F_2(x) \rightarrow F_3(x))) \\ \textit{'some'} \;\mapsto \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \exists x (F_1(x) \wedge F_2(x) \wedge F_3(x)) \end{array} \quad , \quad \begin{array}{c} \dfrac{\text{restaurant}^{\boxed{\texttt{NN}}}}{N} \\ \boxed{\texttt{NN}} \;\mapsto \lambda E \lambda x.\, E(x) \\ \end{array} \tag{2.6}$$

Furthermore, this is by far not the only situation where POS tags are not sufficiently informative on the semantic dimension. Among many other cases of ambiguity, POS tags do not distinguish between coordinating conjunctions nor commas with different semantic functions; and they also fail to differentiate between auxiliary verbs and content verbs or intersective and subsective adjectives [AB17].

Finally, another important aspect when considering lexical semantics is being able to differentiate between different types of Named Entities (NE). This is because one might want to assign different lexical semantics depending on whether such entities are persons, organizations, locations, geo-political entities, events, artifacts, etc. Under POS tagging, all proper nouns are either assigned to the tag NNP or the tag NNPS, without any other useful fine-grained distinctions.

The Univesal Semantic Tagset and the task of universal semantic tagging [AB17] help overcome the shortcomings of POS tags by suggesting a set of semantic tags (or sem-tags) that are agnostic to the syntax and morphology of any given language and concerned only with the meaning contribution of their associated tokens in the spirit of the principle of compositional semantics.

2.2.1. The Universal Semantic Tagset

The Univesal Semantic Tagset [AB17] differs from any POS tagset because it is semantically rich, data-driven, and not tied to a particular syntax. It was originally motivated via the Parallel Meaning Bank (PMB) project, where it currently contributes to the cross-lingual projection of semantic representations [ABE+17].

The most recent version of the universal semantic tagset is depicted in Table 2.2. Similarly to the Penn Treebank tagset [MMS93], we can find closed tag classes and open tag classes. In particular, the coarse semantic tags ATT, COM, NAM, EVE and UNE cover both open and closed class tags, while the rest of the coarse tags cover mainly closed tag classes. The semantic tags that model closed class tokens both disambiguate typically highly ambiguous words and agglutinate cross-lingual spelling variants.

ANA anaphoric	PRO	anaphoric & deictic pronoun: *I, her*
	DEF	definite: *the*
	HAS	possessive pronoun: *my, her*
	REF	reflexive/reciprocal pronoun: *each␣other*
	EMP	emphasizing pronoun: *himself*
ACT speech act	GRE	greeting & parting: *hi, bye*
	ITJ	interjection & exclamation: *alas, ah*
	HES	hesitation: *err*
	QUE	interrogative: *who, which, ?*
ATT attribute	QUC	concrete quantity: *two, six␣million, twice*
	QUV	vague quantity: *millions, many, enough*
	COL	colour: *red, crimson, light␣blue*
	IST	intersective: *open, vegetarian, quickly*
	SST	subsective: *skillful surgeon, tall kid*
	PRI	privative: *former, fake*
	DEG	degree: *2 meters tall, 20 years old*
	INT	intensifier: *very, much, too, rather*
	REL	relation: *in, on, 's, of, after*
	SCO	score: *3-0, grade A*
COM comparative	EQU	equative: *as tall as John, whales are mammals*
	MOR	comparative positive: *better, more*
	LES	comparative negative: *less, worse*
	TOP	superlative positive: *most, mostly*
	BOT	superlative negative: *worst, least*
	ORD	ordinal: *1st, 3rd, third*
UNE unnamed entity	CON	concept: *dog, person*
	ROL	role: *student, brother, prof., victim*
	GRP	group: *John {,} Mary and Sam gathered*
DXS deixis	DXP	place deixis: *here, this, above*
	DXT	temporal deixis: *just, later, tomorrow*
	DXD	discourse deixis: *latter, former, above*
LOG logical	ALT	alternative & repetition: *another, again*
	XCL	exclusive: *only, just*
	NIL	empty semantics: *{.}, to, of*
	DIS	disjunction & exist. quantif.: *a, some*
	IMP	implication: *if, when, unless*
	AND	conjunction & univ. quantif.: *every, and*
MOD modality	NOT	negation: *not, no, neither, without*
	NEC	necessity: *must, should, have to*
	POS	possibility: *might, could, perhaps, can*
DSC discourse	SUB	subordinate relation: *that, because*
	COO	coordinate relation: *so, {,}, {;}, and*
	APP	appositional relation: *{,}, which*
	BUT	contrast: *but, yet*
NAM named entity	PER	person: *Axl␣Rose, Sherlock␣Holmes*
	GPE	geo-political entity: *Paris, Japan*
	GPO	geo-political origin: *Parisian, French*
	GEO	geographical location: *Alps, Nile*
	ORG	organization: *IKEA, EU*
	ART	artifact: *iOS␣7*
	HAP	happening: *Eurovision␣2017*
	UOM	unit of measurement: *meter, $, %*
	CTC	contact information: *112, me@rug.nl*
	URL	URL: http://pmb.let.rug.nl
	LIT	literal usage of a name: *his name is John*
	NTH	undefined name: *table 1a, equation (1)*
EVE events	EXS	untensed simple: *to walk, is eaten*
	ENS	present simple: *we walk, he walks*
	EPS	past simple: *ate, went*
	EXG	untensed progressive: *is running*
	EXT	untensed perfect: *has eaten*
TNS tense & aspect	NOW	present tense: *is skiing, do ski, has skied*
	PST	past tense: *was baked, had gone, did go*
	FUT	future tense: *will, shall*
	PRG	progressive: *has been being treated*
	PFT	perfect: *has been going/done*
TIM temporal entity	DAT	full date: *27.04.2017, 27/04/17*
	DOM	day of month: *27th December*
	YOC	year of century: *2017*
	DOW	day of week: *Thursday*
	MOY	month of year: *April*
	DEC	decade: *80s, 1990s*
	CLO	clocktime: *8:45␣pm, 10␣o'clock, noon*

Table 2.2: Universal Semantic Tagset version 0.7 containing 73 semantic tags grouped into 13 meta-tags, as presented in [AB17].

In principle, semantic tags provide semantic information that treats adjectives and adverbs uniformly and is disjunct from thematic roles, syntax and lemma. Due to this level of abstraction, semantic tags are suitable for cross-lingual applications. We show that we can use semantic tags to define semantic templates [MMMB] in Chapter 5. For now, we will just show in Equation 2.7 that it is straight-forward to rewrite semantic rules for the previously problematic Equation 2.6 using semantic tags.

$$\begin{array}{ccc} \dfrac{\text{any}^{\boxed{\texttt{AND}}} \; / \; \text{some}^{\boxed{\texttt{DIS}}}}{NP/N} & , & \dfrac{\text{restaurant}^{\boxed{\texttt{CON}}}}{N} \\ \boxed{\texttt{AND}} \mapsto \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \forall x(F_1(x) \rightarrow (F_2(x) \rightarrow F_3(x))) & & \boxed{\texttt{CON}} \mapsto \lambda E \lambda x.\, E(x) \\ \boxed{\texttt{DIS}} \mapsto \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \exists x(F_1(x) \wedge F_2(x) \wedge F_3(x)) & & \end{array} \tag{2.7}$$

2.3. Tokenization and other challenges

In general, we assume that a tokenization process takes places before any tagging task. Up until this point, we have been using the term token as a synonym for a meaningful atom that is associated to a tag. However, and specially for semantic tagging, choosing a method for tokenizing the input data should be a matter of special consideration.

One of the main challenges in tokenization is handling Multi-Word Expressions (MWE), that is complex lexical units that expand over multiple words but where the properties of the compound do not decompose over the components. If we consider expressions such as *ice cream* or *kick the bucket*, the compound should only be assigned a single tag corresponding to the associated noun or action, rather than different tags for each word. Thus, MWEs should be identified and accounted for prior to or during the tagging process, and this is a problem to which we will come back in Chapter 4.

Yet another challenge for tagging models comes in the form of unknown words. Unknown words are words that either do not have an associated rule (in rule-based approaches) or do not occur in the training data (in statistical approaches). As discussed in Chapter 3, one needs a pragmatical solution to deal with them such as smoothing techniques or a default tag to which such words can be mapped.

3. Tagging models

In order to formally discuss tagging problems, we need to first convene to a mathematical notation. For the remaining part of this thesis, we will refer to a sequence of n untagged tokens as w_1^n, equivalent to $w_1, ..., w_n$. In a similar fashion, a sequence of n tags will be written as either t_1^n or $t_1, ..., t_n$. A potentially empty subsequence spanning from the i-th to the j-th element of any sequence will be represented as w_i^j or t_i^j.

This thesis generally assumes that the number of possible tokens and tags in a training set is defined by a closed vocabulary and a tagset, which we will represent with the letters $\mathcal{V}$ and $\mathcal{T}$ respectively. Thus, for any given sequence of n tokens, it holds that $w_1^n \in \mathcal{W}^n$; and for any given sequence of tags we have that $t_1^n \in \mathcal{T}^n$.

As we mentioned in the preceding Chapter 2, the goal of problems within the tagging family is to find a sequence of tags $\hat{t}_1^n$ that matches a given sequence of words w_1^n such that the posterior probability $P(\hat{t}_1^n \mid w_1^n)$ is as high as possible:

$$\hat{t}_1^n = \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, P(t_1^n \mid w_1^n) \tag{3.1}$$

Many techniques have been applied in order to solve these problems. Statistical taggers try to learn probability distributions from a training corpus and use them to find the most likely tag sequence. Rule-based taggers, as the name suggests, use a set of rules to assign plausible tags. It is out of the scope of this work to provide an accurate description of them all, and what we will give instead is sufficient background to understand the scope of the experiments performed in the subsequent chapters and their implications.

The reader will note that we will often discuss Part-Of-Speech (POS) taggers, despite POS tagging not being a central topic in this thesis. This is due to the long-sustained attention that POS tagging has enjoyed on behalf of the research community [JM09], and the fact that there are virtually no published results on universal semantic tagging. Thankfully, POS tagging and universal semantic tagging are nearly identical problems from a computational perspective, which means that successful techniques in one task are easily transferable to the other and contrariwise.

3.1. Discriminative and generative models

Human languages typically contain a large number of words and constructions. Because of this variability, one realizes that it is impossible to accurately compute the posterior probability $P(t_1^n \mid w_1^n)$ in Equation 3.1. Thus, statistical tagging models typically make some simplifying assumptions. We will distinguish between models that attempt to estimate the posterior probability directly, also known as discriminative models; and models that estimate the likelihood and prior probabilities instead, also known as generative models.

The reasoning behind discriminative models starts by considering the definition of conditional probability, which allows us to write:

$$P(t_1^n \mid w_1^n) = \frac{P(t_1^n, w_1^n)}{P(w_1^n)} = \frac{\prod_{i=1}^{n} P(w_1^n, t_1^i)}{\prod_{i=1}^{n} P(w_1^n, t_1^{i-1})} \tag{3.2}$$

Furthermore, by using the chain rule of conditional probability and grouping the factors present in Equation 3.2, we arrive at the following:

$$P(t_1^n \mid w_1^n) = \prod_{i=1}^{n} P(t_i \mid w_1^n, t_1^{i-1}) \tag{3.3}$$

Equation 3.3 does not present a problem any simpler than Equation 3.2. However, the formula is presented in a manner that allows for some independence assumptions which make computing $P(t_1^n \mid w_1^n)$ as the product of approximable quantities feasible. Traditionally, two assumptions are made in tagging problems. The first one is that each t_i is only dependent on w_i and independent with respect to all other tokens. The second assumption is that t_i is only dependent on a certain number of previous tags, and we refer to models that condition on the most recent $k-1$ tags k-gram models.

$$P(t_i \mid w_1^n, t_1^{i-1}) \approx P(t_i \mid w_i, t_1^{i-1}) \tag{3.4}$$

$$P(t_i \mid w_1^n, t_1^{i-1}) \approx P(t_i \mid w_1^n, t_{i-k+1}^{i-1}) \tag{3.5}$$

Finally, we can substitute Equation 3.4 and Equation 3.5 into Equation 3.3 to obtain a product of terms representing the conditional probabilities of each tag given their corresponding token and previous tags. Such terms can be approximated from a labeled corpus using techniques such as Maximum Likelihood Estimation (MLE) or Expectation Maximization (EM).

$$\hat{t}_1^n = \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, P(t_1^n \mid w_1^n) = \prod_{i=1}^{n} P(t_i \mid w_i, t_{i-k+1}^{i-1}) \tag{3.6}$$

The approach chosen by generative models is slighly different than the one of discriminative models. Considering again Equation 3.1, we can rewrite it using Bayes rule as shown in Equation 3.7. Note that the normalization constant $P(w_1^n)$ can be discarded since it does not affect the result of our maximization problem:

$$\begin{aligned} \hat{t}_1^n &= \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, P(t_1^n \mid w_1^n) \\ &= \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)} \\ &\propto \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, P(w_1^n \mid t_1^n)P(t_1^n) \end{aligned} \tag{3.7}$$

It is once again unfeasible to estimate the likelihood and prior probabilities $P(w_1^n \mid t_1^n)$ and $P(t_1^n)$ directly due to the dimensionality of the associated data. By adopting the same independence assumptions which were presented in Equation 3.4 and Equation 3.5, and further assuming that the probability of a token appearing at a certain position is only dependent on the associated tag, we are able to write the following:

$$P(w_1^n \mid t_1^n) = \prod_{i=1}^{n} P(w_i \mid t_i) \tag{3.8}$$

$$P(t_1^n) = \prod_{i=1}^{n} P(t_i \mid t_{i-k+1}^{i-1}) \tag{3.9}$$

Having empirically defined the likelihood and prior probabilities, the only remaining step is to substitute these definitions into Equation 3.7 to arrive at the descriptive formula for generative models shown in Equation 3.10.

$$\begin{aligned}
\hat{t}_1^n &\propto \underset{t_1^n \in \mathcal{T}^n}{\arg\max}\, P(w_1^n \mid t_1^n) P(t_1^n) \\
&= \underset{t_1^n \in \mathcal{T}^n}{\arg\max} \prod_{i=1}^{n} P(w_i \mid t_i) \prod_{i=1}^{n} P(t_i \mid t_{i-k+1}^{i-1}) \\
&= \underset{t_1^n \in \mathcal{T}^n}{\arg\max} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-k+1}^{i-1})
\end{aligned} \tag{3.10}$$

3.2. Most Frequent Class taggers

The main difficulty of a tagging problem such as POS tagging is finding a method for resolving the ambiguity that arises when a given surface form of a token can be assigned to different tags depending on its context. Most surface forms such as proper nouns are not ambiguous, but the ambiguous forms typically represent some of the most common words in a language. In the case of English, between 55% and 67% of the tokens in a text are ambiguous, as shown in Table 3.1.

	WSJ	**Brown**
Surface forms		
Unambiguous (1 tag)	44,432 (86%)	45,799 (85%)
Ambiguous ($\geq$ 2 tags)	7,025 (14%)	8,050 (15%)
Tokens		
Unambiguous (1 tag)	577,421 (45%)	384,349 (33%)
Ambiguous ($\geq$ 2 tags)	711,780 (55%)	786,646 (67%)

Table 3.1: The amount of tag ambiguity for token surface forms and tokens in a POS-tagged version of the WSJ corpus and the Brown corpus [FK79], adapted from [JM09].

Nonetheless, the distribution of possible tags linked to a particular surface form is far from uniform, with some tags being much more frequent than others. One could entertain the idea of a simple baseline approach that merely maps each token to the tag with which it is associated most often in the training data. These models are called Most Frequent Class (MFC) taggers and, despite their simplicity, obtain tagging accuracies ranging from 80% to 90% depending on the dataset and language [ID10]. Thus, when assessing the performance of more advanced models, it is important to consider taggers such as MFC in order to gain some perspective on the significance of the evaluation metrics.

3.3. Transformation-Based Learning

Early attempts at formulating POS tagging models employed a set of handcrafted rules together with the vocabulary $\mathcal{W}$ in order to specify the conditions under which tags could be assigned to tokens [ID10]. However, this required both extensive manual work and linguistic knowledge, which made the approach impracticable in many situations.

Back in those days, a particularly revolutionary approach known as Transformation-Based Learning (TBL) allowed for automatically learning the required linguistic rules [Bri95]. In short, the algorithm assigns a tag to every token in the training corpus either by using a rule from a set of known rules or randomly when no applicable rules exist. Then, the algorithm tries to induce the template-based rule capable of correcting the most mistakes, adds the rule found to the set of known rules and tags the training corpus again. New rules keep being found until the error rate is not reduced anymore or until it falls below a predefined threshold.

The TBL tagger stands the test of time and still today poses some advantages over more modern statistical and neural approaches. This is the case because the rule templates used can virtually utilize any source of information available to them, and the trained model is easily interpretable by humans and not bound to overfitting. The TBL tagger was trained and tested on the Wall-Street Journal (WSJ) corpus, achieving an accuracy of 96.6% and learning 690 different rules.

3.4. Hidden Markov Models

Hidden Markov Models (HMM) constitute a widely studied statistical formalism for POS tagging and fall under the umbrella of generative models described by Equation 3.10. When considering a tagging problem, one can only observe the tokens of a given phrase directly, which are generated by the emission probabilities $P(w_i \mid t_i)$. Thus, the tags associated to the tokens become hidden (non-observable) states of the model and the terms $P(t_i \mid t_{i-1}...t_{i-k+1})$ denote the transition probabilities between them.

Training a HMM entails maximizing the probability of generating an observed sequence of tokens. It is possible to approximate both the emission and transition probabilities using MLE and a tagged corpus as shown in Equation 3.11. Note that, in our formulas, $C(w_i, t_i)$ denotes the number of times a word w_i is labeled with the tag t_i. Similarly, $C(t_i^{i+x})$ represents the number of occurrences of the tag sequence t_i^{i+x}.

$$P(w_i \mid t_i) = \frac{C(w_i, t_i)}{C(t_i^i)} \qquad P(t_i \mid t_{i-1}...t_{i-k+1}) = \frac{C(t_{i-k+1}^i)}{C(t_{i-k+1}^{i-1})} \tag{3.11}$$

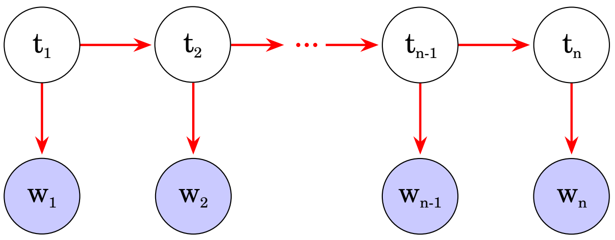

Figure 3.1: Visual representation of a first order HMM defining a joint probability $P(w_1^n, t_1^n)$. Transition probabilities are only conditioned on the previous hidden state ($k = 2$).

However, finding the most likely sequence of tags associated with an observed sequence is a computationally complex problem, since there might be many possible state sequences (or paths) that generate the observed tokens. In general, one is interested in obtaining the path with the highest probability $\hat{\pi}$ as defined in Equation 3.12.

$$P(w_1^n \mid \hat{\pi}) = \arg\max_{\pi} P(w_1^n \mid \pi) \tag{3.12}$$

The Viterbi algorithm [JM09] is a dynamic programming procedure capable of computing the path $\hat{\pi}$. Assuming a starting state s and an ending state e, it computes the joint path and emission probability as $V(n, e)$, defined recursively in Equation 3.13. Furthermore, we can build the path π^* after all $V(i, j)$ quantities have been computed by tracing back our choices from $V(n, e)$ to $V(1, s)$.

$$\begin{aligned} &\textit{Initialization:} \quad && V(1, s) = 1 \\ &\textit{Recursion:} \quad && V(i, j) = \max_{l} V(i-1, l) P(t_j \mid t_{j-k+1}^{l}) P(w_i \mid t_j) \end{aligned} \tag{3.13}$$

The *TnT* tagger is a system that employs the HMM formulation using a context window of $k = 3$ tokens by default [Bra00]. This tagger also features other enhancements, such as smoothing of the estimated prior probabilities using linear interpolation and the use of a differentiated set of tags for uppercase and lowercase words. To increase the efficiency of the tagger, beam search is used in conjunction with the Viterbi algorithm to prune non-promising search paths while scanning a given sentence. The *TnT* tagger achieves an accuracy of 96.7% on the Penn Treebank [MMS93].

3.5. Conditional Random Fields

The HMM architecture presented in the previous section is effective in the POS tagging task [CKPS92], but some issues with the formalism exist. The most important shortcoming is that HMMs only capture dependences between each tag and its corresponding token, failing to take potentially informative previous and later tokens into account. Furthermore, HMMs learn the joint distribution of tags and tokens $P(w_1^n, t_1^n)$, while in tagging tasks it would be more convenient to learn the conditional probability $P(t_1^n \mid w_1^n)$ directly.

A Conditional Random Field (CRF) is an undirected probabilistic graphical model with nodes denoting random variables and edges denoting dependencies between those variables [LMP01]. They are an example of discriminative models defined by Equation 3.3 capable of modeling the dependence between each tag and the entire sequence of tokens. Instances of CRFs which only model the dependency between neighboring tags directly are called linear chain CRFs and are typically used in tagging instead of more general forms [JM09].

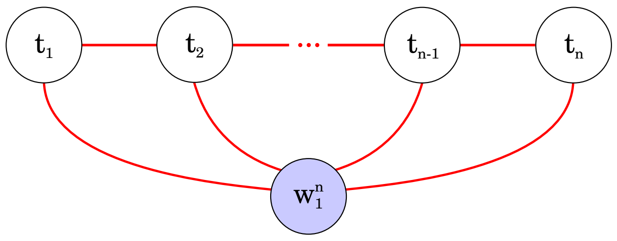

Figure 3.2: Visual representation of a linear chain CRF defining a conditional probability $P(t_1^n \mid w_1^n)$. The entire observed sequence is available at any given state of the model.

We can define the distribution $P(t_1^n \mid w_1^n)$ as shown in Equation 3.14 [LMP01]. In our formulation, $Z(w_1^n)$ is a normalization constant, ϕ denotes the set of feature functions, and θ represents the set of model parameters which need to be estimated in training time.

$$\begin{aligned} P(t_1^n \mid w_1^n; \theta) &= \frac{1}{Z(w_1^n)} \prod_{i=1}^{n} \exp\left(\sum_{k=1}^{K} \theta_k \phi_k(t_i, t_{i-1}, w_1^n, i) \right) \\ &\propto \exp\left(\sum_{k=1}^{K} \theta_k \sum_{i=1}^{n} \phi_k(t_i, t_{i-1}, w_1^n, i) \right) \end{aligned} \tag{3.14}$$

The parameters of a CRF model can be computed using MLE by considering a training set $D = \{(W_i, T_i) : i = 1, \ldots, N\}$ where each pair (W_i, T_i) contains a sequence of tokens and its corresponding sequence of tags. The objective function shown in Equation 3.15 follows, where the penalty incurred by a training example is the negative log-probability of the correct tag sequence and a regularization factor [GRDB07]. Then, one can employ the back-propagation algorithm in order to obtain a set of optimal parameters [Hay98].

$$\mathcal{L}(\theta, D) = -\log\left(\prod_{i=1}^{N} P(T_i \mid W_i; \theta) + \frac{C}{2} \|W\|^2 \right) \tag{3.15}$$

Finally, the decoding process on a CRF is simply the problem of finding a sequence of tags that maximizes the conditional probability $P(t_1^n \mid w_1^n)$ given the set of optimal parameters. This search can be solved efficiently using the Viterbi algorithm as in the case of HMMs. Approaches that employed CRFs for POS tagging report accuracy scores revolving around 97.0% for English on the Penn Treebank [SRLK14].

3.6. Artificial Neural Networks

Following important technological developments, Artificial Neural Networks (ANN) have become a wide-spread representation learning technique which has given birth to some of the best-performing systems in various areas. However, neural networks have not consolidated as a popular modeling choice for POS tagging, possibly because of the not entirely favorable trade-off between added complexity and performance improvement.

ANNs are created by combining a number of basic components called perceptrons (or units). A perceptron typically has an input vector $\mathbf{x} \in \mathbb{R}^l$, a weight vector $\mathbf{w} \in \mathbb{R}^l$, a bias term $b \in \mathbb{R}$, and an activation function f. The only duty that these units perform is to compute an output $\hat{y} \in \mathbb{R}$ as a squashed weighted sum of the input vector:

$$\hat{y} = f\left(\mathbf{wx} + b\right) \tag{3.16}$$

Moreover, the output of a perceptron can be turned into a component of the input vector of another perceptron. By considering a collection of ordered disjoint sets of perceptrons and making the output of each perceptron in a given set become an input for each perceptron in the next set, one can construct a type of feed-forward network known by the name of Multi-Layer Perceptron (MLP) or simply deep neural network.

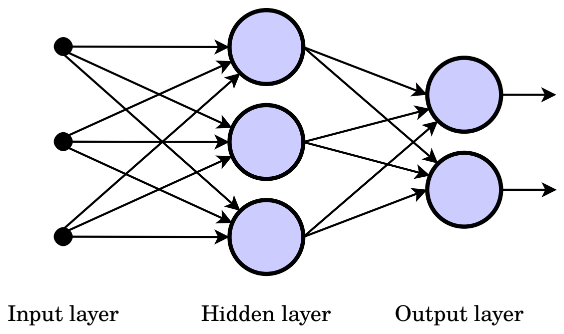

Figure 3.3: Visual representation of an MLP containing 3 layers. The output of each perceptron in a given layer is connected to all perceptrons in the next layer.

.

An MLP can approximate any continuous multivariate function to any degree of accuracy, provided there are sufficiently many hidden perceptrons [HSW89]. Thus, it is possible to make MLPs learn complex functions by selecting the appropriate values for the weight and bias terms in each contained perceptron. This can be achieved using a process known as backpropagation [Gur97], which iteratively presents training input examples to the network and computes the resulting output. The mismatch observed between the output and the desired value is propagated backwards in order to adjust the network parameters via gradient descent according to Equation 3.17 and Equation 3.18.

In our formulation, $\mathbf{W}^{(t-1)} \in \mathbb{R}^{n \times n}$ is the previous weight matrix and $\mathbf{W}^{(t)} \in \mathbb{R}^{n \times n}$ the updated weight matrix at a time step t. The same superscript notation applies to the bias vector $\mathbf{b} \in \mathbb{R}^n$. These values are updated according to the value of their gradient with respect to a loss function E and the quantity $\gamma(t)$, known as the learning rate. The interested reader can check the entire derivation from a more detailed source [Hay98].

$$\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} - \gamma(t) \cdot \left[\frac{\partial E}{\partial w_{ij}} \right]_{w_{ij} \in \mathbf{W}^{(t-1)}} \tag{3.17}$$

$$\mathbf{b}^{(t)} = \mathbf{b}^{(t-1)} - \gamma(t) \cdot \left[\frac{\partial E}{\partial b_i} \right]_{b_i \in \mathbf{b}^{(t-1)}} \tag{3.18}$$

One of the earliest attempts at using an MLP for POS tagging is the *Net-tagger* system [Sch94], which uses a wide-range of information sources such as the surface form of tokens and their context as inputs to the network. In contrast, each unit in the output layer is responsible for outputting the likelihood of a single POS tag. The results presented show an accuracy improvement of approximately 2% over HMM models in the particular dataset employed. It is worth mentioning that *Net-tagger* has inspired many similar tagging architectures in languages other than English [Jan04, Bo08].

3.6.1. Recurrent Neural Networks

We will now consider another type of neural networks known as Recurrent Neural Networks (RNN), which are able to capture the dynamics of sequences present in tagging problems through loops in their structure [Lip15]. At each time step t, a hidden unit of an RNN takes an input vector $\mathbf{x}^{(t)} \in \mathbb{R}^n$ and the output of the previous hidden unit $\mathbf{h}^{(t-1)} \in \mathbb{R}^m$. The next hidden state $\mathbf{h}^{(t)}$ is then computed by applying the following recursive operation, where $\mathbf{W} \in \mathbb{R}^{m \times n}$, $\mathbf{U} \in \mathbb{R}^{m \times m}$, and $\mathbf{b} \in \mathbb{R}^m$ are the weight and bias parameters respectively, and f is the activation function of the hidden unit [KJSR16].

$$\mathbf{h}^{(t)} = f(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{b}) \tag{3.19}$$

A hidden state $\mathbf{h}^{(t)}$ as currently formulated should be able to summarize all historical information up to the time step t. However, this is not the case in practice, since numerical approximation problems can make the transfered gradient value vanish or explode in long sequences [BSF94]. This problem motivated the creation of gating units that replace the function f and enable RNNs to memorize information even in sequences spanning over a large number of time steps.

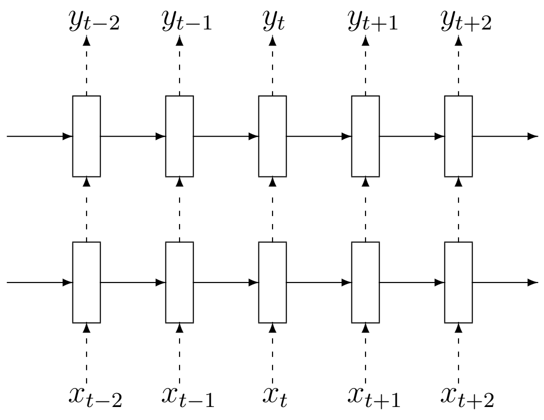

Figure 3.4: Visual representation of a multi-layer RNN extracted from [ZSV14]. Each hidden unit receives the output of the current input unit and the previous hidden unit, and sends information to the next hidden unit and the hidden unit in the upper layer.

3.6.2. Gating mechanisms

Long Short-Term Memory (LSTM) units are an extensively studied gating mechanism that allows RNNs to avoid the vanishing gradient problem [HS97]. An LSTM unit introduces a cell state $\mathbf{c} \in \mathbb{R}^n$ and computes the values $\mathbf{h}^{(t)}$ and $\mathbf{c}^{(t)}$ at each time step t by taking into account the input vector $\mathbf{x}^{(t)}$, the output of the previous hidden unit $\mathbf{h}^{(t-1)}$, and its previous cell state $\mathbf{c}^{(t-1)}$. The interaction is described in detail in Equation 3.20, where the terms $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$, $\mathbf{o}^{(t)}$ are referred to as input, forget and output gates respectively, the symbols σ and ϕ denote the application of element-wise sigmoid and hyperbolic tangent functions, and $\odot$ indicates element-wise multiplication [KJSR16].

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{W}^i \mathbf{x}^{(t)} + \mathbf{U}^i \mathbf{h}^{(t-1)} + \mathbf{b}^i) \\
\mathbf{f}^{(t)} &= \sigma(\mathbf{W}^f \mathbf{x}^{(t)} + \mathbf{U}^f \mathbf{h}^{(t-1)} + \mathbf{b}^f) \\
\mathbf{o}^{(t)} &= \sigma(\mathbf{W}^o \mathbf{x}^{(t)} + \mathbf{U}^o \mathbf{h}^{(t-1)} + \mathbf{b}^o) \\
\mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \phi(\mathbf{W}^g \mathbf{x}^{(t)} + \mathbf{U}^g \mathbf{h}^{(t-1)} + \mathbf{b}^g) \\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \phi(\mathbf{c}^{(t)})
\end{aligned} \tag{3.20}$$

Furthermore, it is possible to extend both the vanilla RNN and LSTM architectures by adding hidden units in additional layers. The procedure is similar to the process of building MLPs and simply makes the units in the next layer accept the input $\mathbf{h}^{(t)}$ from the previous layer at each time step t, as shown in Figure 3.4. Nowadays, these deep architectures are necessary in order to obtain state-of-the-art performance in various tasks [PGCB13].

Aside from LSTMs, other typologies of gating mechanisms exist and are used in the literature. The most notorious ones are Gated Recurrent Units (GRU) [CvMG+14], which are characterized by a reset gate $\mathbf{r}^{(t)}$ and an update gate $\mathbf{z}^{(t)}$. Intuitively, these two gates attempt to control how much information from the previous hidden state and the current input is taken into account. GRUs proceed using the following transformations:

$$\begin{aligned}
\mathbf{r}^{(t)} &= \sigma(\mathbf{W}^r \mathbf{x}^{(t)} + \mathbf{U}^r \mathbf{h}^{(t-1)}) \\
\mathbf{z}^{(t)} &= \sigma(\mathbf{W}^z \mathbf{x}^{(t)} + \mathbf{U}^z \mathbf{h}^{(t-1)}) \\
\mathbf{h}^{(t)} &= (1 - \mathbf{z}^{(t)}) \odot \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \odot \phi(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}(\mathbf{r}^{(t)} \odot \mathbf{h}^{(t-1)}))
\end{aligned} \tag{3.21}$$

3.6.3. Bidirectionality

We have seen that LSTM networks can efficiently capture past information within an arbitrary long sequence and use that information at any particular time step. However, in tagging tasks, the information provided by future tokens and tags in a given phrase can be as important as the past history. In a manner remarkably similar to HMMs, the LSTM networks described up to this point fail to incorporate future events for modeling, since the information always flows linearly from one time step to the next and never in the opposite direction.

Nonetheless, it is possible to design a network that can capture both past and future information by considering two separate LSTM layers. While having both LSTM layers connected to the input and the output layers, one can make the information in them flow in opposite directions and create a Bidirectional LSTM network (BLSTM) [SP97]. Thus, the first layer incorporates the recurrent connections from past time steps, and the second layer incorporates the recurrent connections from future time steps. Finally, the output of the network can be computed as shown in Equation 3.22, where $\mathcal{S}$ denotes the output activation function and $\oplus$ indicates element-wise addition or an otherwise combining operation.

$$\hat{y}^{(t)} = \mathcal{S}(\mathbf{W}_{\overleftarrow{h}}\overleftarrow{h}^{(t)} \oplus \mathbf{W}_{\vec{h}}\vec{h}^{(t)} + \mathbf{b}_{\hat{y}}) \tag{3.22}$$

The effectiveness of BLSTMs together with word embeddings for POS tagging has been researched by [PSG16], concluding that they are superior to HMM and CRF models for English and a variety of other languages. Other similar studies include a variation of the BLSTM model which features a CRF in the output layer replacing the output activation function as shown in Figure 3.5, and obtain accuracy scores of approximately 97.5% using their respective training and test datasets [HXY15, MH16].

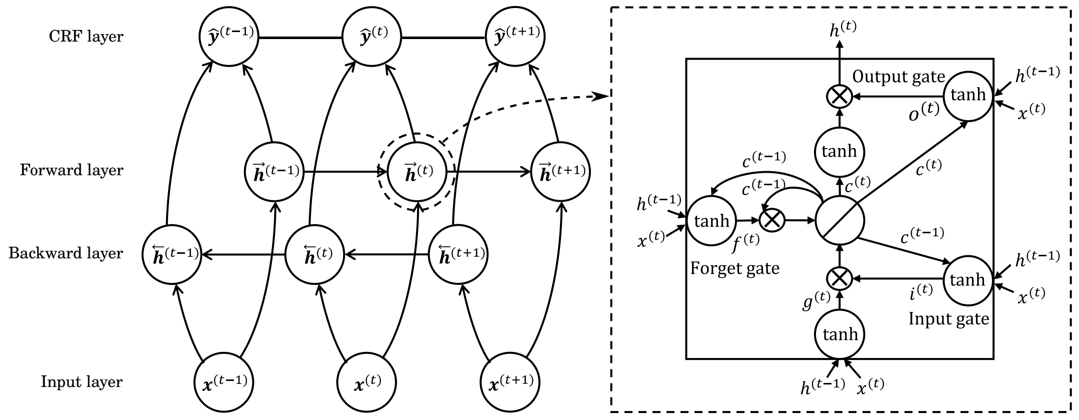

Figure 3.5: Visual representation of a BLSTM neural network featuring a CRF in its output layer. The block on the right illustrates the structure of each hidden LSTM unit.

The model in [BPB16] further considers applying BGRU networks to the task of universal semantic tagging. The structure of their suggested model is hierarchical and employs both GRUs at an upper level and residual networks at a lower level [HZRS16]. While the version of the Universal Semantic Tagset used differs from the one presented in Section 2, this model achieves an accuracy of 83.64% when evaluated on the corresponding gold test data. The underlying implication appears to be that, currently, universal semantic tagging is a far more challenging task than POS tagging.

3.6.4. Residual Networks

It is often the case that deeper ANN models outperform shallower alternatives. However, it has also been observed that one cannot simply keep adding layers since at some point the training accuracy gets saturated and decreases. This numerical impossibility of learning an optimal mapping in very deep networks is also know as the degradation problem, an it is addressed by a neural architecture known as Residual Network (ResNet) [HZRS16].

Instead of making all layers in a neural network jointly learn a function $\mathbf{y} = \mathcal{G}(\mathbf{x})$, one can define a residual function $\mathcal{F}(\mathbf{x})$ such that $\mathbf{y} = \mathcal{F}(\mathbf{x}) + \mathbf{W}^{\mathcal{F}}\mathbf{x}$ for some parameters $\mathbf{W}^{\mathcal{F}}$. ResNets work under the hypothesis that the residual function $\mathcal{F}$ is easier to optimize than the global function $\mathcal{G}$ by stacking residual blocks like the one shown in Figure 3.6.

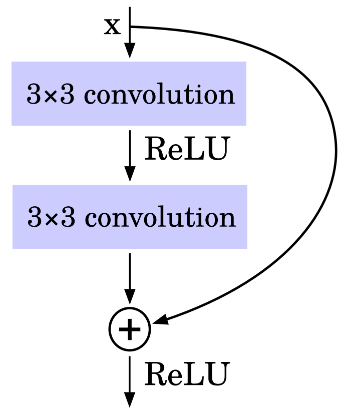

Figure 3.6: Visual representation of a residual block containing two convolutional operations using a 3×3 filter and a shortcut representing the identity function.

In this thesis, we will only consider residual blocks based on the identity mapping and where the input and output dimensions are the same, which can be defined without the need of additional parameters by Equation 3.23. Here, we employ term $\mathbf{x}_i$ to represent the input of a residual block and the term $\mathbf{y}_i$ to refer to its output.

$$\mathbf{y}_i = \mathcal{F}_i(\mathbf{x}_i) + \mathbf{x}_i \tag{3.23}$$

The effectiveness of ResNets lies in the experimental evidence that learning a mapping $\mathcal{F}_i(\mathbf{x}_i) \approx 0$ in unnecessary residual blocks is a computationally easier task that learning the unreferenced mapping $\mathbf{y}_i \approx \mathbf{x}_i$. Being able to learn these residual mappings effectively allows ResNets to construct arbitrarily deep architectures and gain maximal representational capabilities without incurring in degradation.

4. Building a semantic tagger

A significant part of the present thesis focuses on building efficient and usable models for the task of universal semantic tagging [AB17]. This is done by drawing on state-of-the-art models for Part-Of-Speech (POS) tagging [HXY15, MH16] and universal semantic tagging [BPB16], since both tasks share a similar formulation from a machine learning perspective and published results are practically nonexistent for the latter. The resulting software tool which we implemented is available from `https://github.com/ginesam/semtagger`, and employs Keras with a Tensorflow backend [C$^+$15] in order to represent neural models that can have its parameters and architecture seamlessly customized.

We will next discuss a wide range of topics that are connected with the implementation of such a tool from a technical perspective, such as the data used, preprocessing steps, feature engineering, parameter optimization and other intrinsicalities. Finally, we compare the performance of different typologies of models and identify some recurrent difficulties in the task of universal semantic tagging.

4.1. The Parallel Meaning Bank

The data that we use to train and test our models comes exclusively from the Parallel Meaning Bank (PMB) [ABE$^+$17], which is a reasonably-sized semantically annotated parallel corpus for English, German, Dutch and Italian. It contains texts in raw and tokenised formats, word senses, thematic roles, and formal meaning representations. Naturally, it also contains tokenized and semantically tagged sentences that can be used for our purpose of training a universal semantic tagger in a supervised manner.

The semantically tagged documents present in the PMB fall within 3 different categories according to the quality of the associated tags: gold, silver and bronze. Gold documents are manually tagged by semanticists and can be considered correct, albeit they might suffer from inter-annotator disagreement. On the other end of the spectrum, bronze documents are automatically tagged using the *TnT* tagger [Bra00]. Silver documents are essentially bronze sentences that include at least one correction made by a human.

Upon inspection, one can easily realize that the bronze data contains many mistakes, rarely comprising any sentences where all semantic tags could be considered correct. The same can be said about the silver data, since a typical silver document contains a single correction. Because of these fundamental differences between different subsets of data, in this thesis we decide to ignore the bronze data completely.

We further randomly split both the gold and silver subsets into training data and test data, as shown in Table 4.1. Ideally, one would like to train a tagging model using only gold training data, but this is infeasible due to the limited number of documents that the gold subset contains. Thus, all tagging models presented in the current chapter of this thesis are jointly trained using the gold and silver training splits, and separately evaluated on each of the gold and silver test splits.

Parallel Meaning Bank (PMB) splits			
Gold data		*Silver data*	
Training	Test	Training	Test
4,894	544	56,465	6,274

Table 4.1: Number of documents in each of our generated splits of the PMB. Both training splits account for approximately 90% of the gold and silver sentences respectively.

4.2. Document length normalization

There are further differences between the gold and silver documents in the PMB. Possibly because manually tagging a document is a laborious task, gold documents tend to be shorter than silver documents and are often comprised of a single sentence. Nonetheless, silver documents usually contain many more sentences in comparison.

It is often the case that, when building recurrent neural models for POS tagging, one limits the length of the input sequences forwarded to the network [BPB16]. This practice attempts to make the tagging problem more consistent and to avoid issues associated with very long sequences such as vanishing gradients, exploding gradients, or excessively dilated training times [CZH+17].

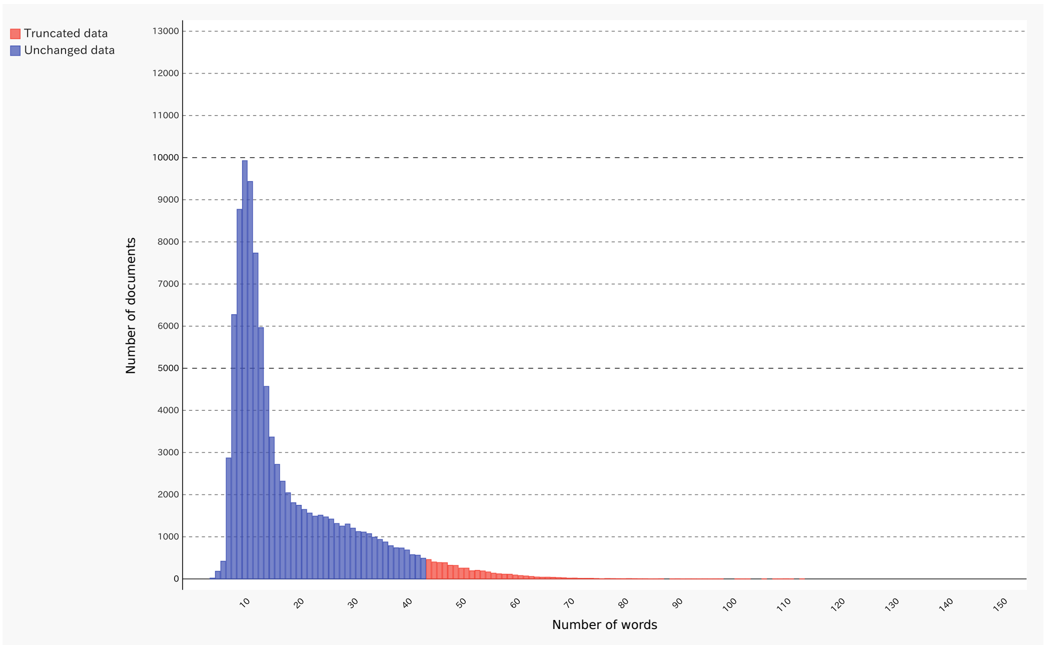

Figure 4.1: Distribution of document lengths in the gold and silver training splits combined after dividing documents into sentences.The results reported in this thesis limit their input length at the percentile 0.95 of this distribution (42 words).

This self-imposed length limitation is problematic because the combined gold and silver training sets of the PMB present a document length distribution with a remarkably elongated tail. Thus, many documents which could contain informative input sequences would be truncated while training our models. In order to make better use of our data, we split the PMB documents into sentences, and consider each resulting sentence as an independent training item. This allows us to increase the number of non-truncated training items from approximately 60,000 (Table 4.1) to over 110,000 (Figure 4.1).

4.3. Feature engineering

Neural models such as the ones that our software provides need to construct a numerical tensor from the input tokens in order to train from the PMB using the backpropagation algorithm. For the purpose of universal semantic tagging, we should find a function that maps tokens to feature tensors such that tokens associated to the same semantic tag are also assigned similar representations.

One possible way of obtaining an approximation to our desired features is by using word embeddings, a technique popular in many natural language processing applications which maps tokens to vector representations and captures several syntactic and semantic properties [MCCD13]. In particular, our software employs pre-trained embedding matrices using the *GloVe* model [PSM14] to construct 2-dimensional token-based representations of the input sentences $F_w \in \mathbb{R}^{l_s \times d_w}$, where l_s represents the maximum sequence length and d_w the dimensionality of the representation. Note that special padding symbols are used when the input sequence contains less than l_s tokens, and that the input sequence is truncated when the number of tokens is greater than l_s.

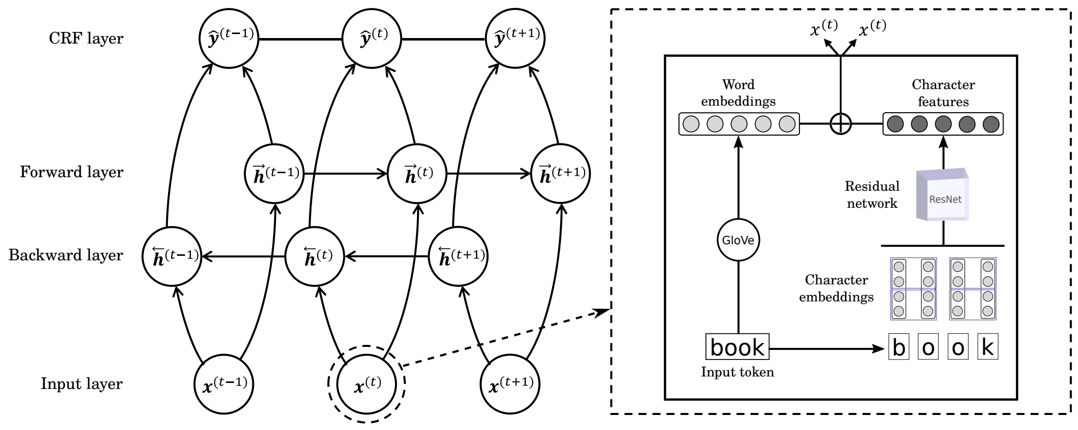

Figure 4.2: Visual representation of a BLSTM neural network featuring a CRF in its output layer. The block on the right illustrates the process of creating an input representation combining both token and character-based features.

Apart from token-based representations, several researchers have also successfully employed sub-token-based representations for POS tagging and universal semantic tagging, proposing hierarchical models that employ bidirectional Recurrent Neural Networks (RNN) or Residual Networks (ResNet) in order to build representations from characters or bytes [PSG16, BPB16]. Our software is also able to provide character-based representations by employing ResNets within an overall hierarchical structure, in a manner inspired by the models in [BPB16].

The procedure for building character-based representations entails defining some randomly initialized character embeddings to construct 3-dimensional character-based representations of the input sequence $F_c \in \mathbb{R}^{l_s \times l_w \times d_c}$, with l_s denoting the maximum sequence length, l_w the maximum length of a word, and d_c the dimensionality of the character embeddings. This representation is then used as the input of a low-level ResNet containing residual blocks as shown in Figure 3.6, which is tasked with learning local features by virtue of the convolution operations performed [DSZ14].

In our system, token and character-based features can be used in isolation or jointly. In the latter case, the convoluted representation is reshaped into a 2-dimensional vector $F_{\hat{w}} \in \mathbb{R}^{l_s \times K}$, and concatenated with F_w. The resulting concatenation is thus a 2-dimensional vector that contains information from both tokens and characters, and is used as input for a high-level neural network in charge of learning the mapping from features to semantic tags. This entire process is illustrated graphically in Figure 4.2.

4.4. Out-Of-Vocabulary tokens

All models that our implemented software tool can be configured to represent are trained under a closed-vocabulary for both tokens and characters, which we will respectively denote with the terms $\mathcal{V}_w$ and $\mathcal{V}_c$. Since we employ pre-trained embeddings in order to featurize input tokens, $\mathcal{V}_w$ is automatically defined by all the words present in the embedding matrix. On the other hand, $\mathcal{V}_c$ is automatically build and contains all individual characters that are present in the training data.

In some cases during the training process or when predicting semantic tags for a given input, one might encounter tokens that are not contained in $\mathcal{V}_w$. We will refer to these tokens as Out-Of-Vocabulary (OOV) tokens. Note that it is also possible to encounter characters that are not contained in $\mathcal{V}_c$, which is a recurrent problem when working with languages such as Japanese or Chinese. However, in the case of English, the set of characters occurring within the training data is typically sufficient to represent nearly the totality of tokens in the language.

OOV tokens pose a problem in models that use word embeddings. Since there is not a correspondence between a given OOV token and an embedding vector, it is challenging to make an accurate tag prediction. Furthermore, OOV tokens negatively affect the tag predictions of the surrounding words and the training procedure. Our system addresses such issues by introducing a special embedding vector that applies to all OOV tokens.

Moreover, there is a particular type of token that must be subject of special consideration when becoming an OOV token. This is the case of Multi-Word Expressions (MWE), which we will define from a practical perspective as groups of words that should be assigned the same semantic tag. In general, our system assumes that MWEs are properly tokenized, allowing users to input either *ice cream* or *ice~cream/ice-cream* depending on whether they want to have one or two semantic tags associated with the expression.

Because of their nature, one can suggest non-linguistically motivated ad-hoc procedures to deal with MWEs. In particular, our system attempts to split them into their constituents and find an embedding for each resulting part, accepting variations of the new token such as lowercased or uppercased versions. If the corresponding embeddings can be found, the constituents are considered as if they were separate tokens. During the prediction phase, the same splitting process takes places when appropriate, and the compound is assigned a sem-tag decided via the majority voting result of the sem-tags of its constituents.

4.5. Semantic tag distribution

Another important aspect that we need to analyze in order to gain more insights on the PMB data and to understand the results later presented in this chapter is the distribution of semantic tags. We present this Zipfian distribution as extracted from the combined gold and silver training data in Figure 4.3. The direct conclusion is that, while some semantic tags are incredibly common, others seldom occur within a sentence.

One can observe that the most frequent sem-tag is `CON`, assigned to all concepts and occurring nearly twice as much as the second most frequent sem-tag `NIL`, which denotes empty semantics and is often assigned to tokens acting as linguistic glue. The sem-tags denoting intersective attributes (`IST`) and universal quantification (`AND`) follow.

On the opposite side of the curve we find sem-tags denoting place deixis (DXP), progressive tense (PRG) and groups of people (GRP), among others connected to specific types of attributes. While a priori we would not consider some of the associated tokens to be extremely infrequent, they might prove challenging to tag for a model such as *TnT* [Bra00] due to their ambiguity. One must keep in mind that a very significant percentage of the silver data is tagged automatically after all.

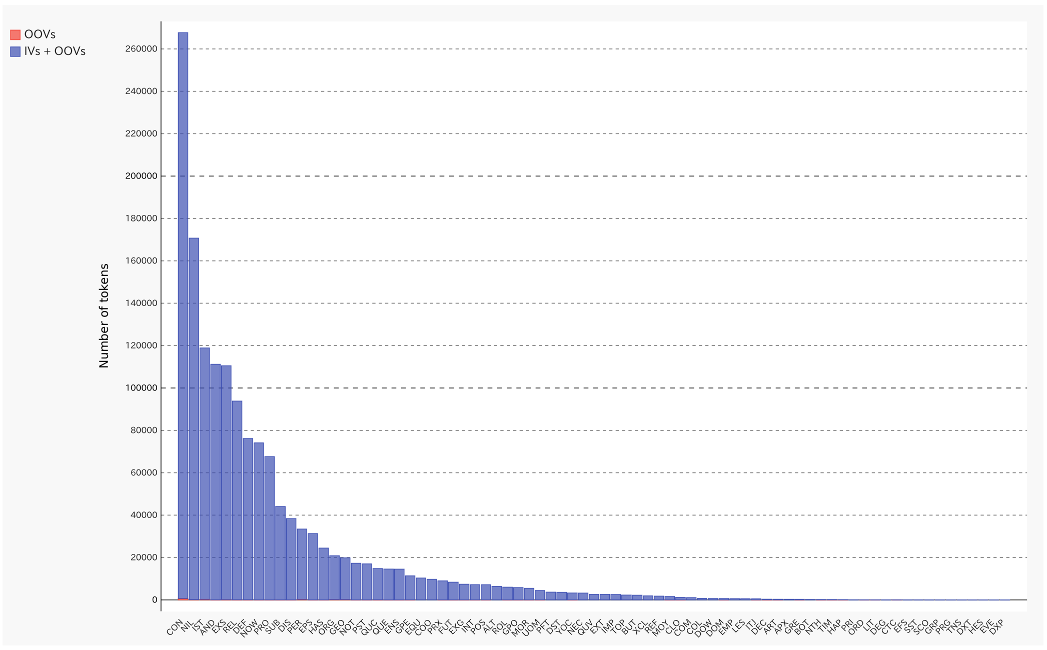

Figure 4.3: Distribution of semantic tags in the gold and silver training splits combined after preprocessing, where approximately half of the possible semantic tags are virtually unused. The corresponding vocabulary is defined by *GloVe* embeddings trained on the Common Crawl corpus [PSM14], and the few semantic tags associated to OOV words are shown in red.

Having such disparity between tags poses yet another challenge for estimating a neural model. This is because neural models rely on large amounts of data and repeated presentation of the same information in order to effectively use gradient descent. In such a setting as the PMB, it might prove challenging to capture the information associated to the least frequent semantic tags. In fact, it is likely that trained models confuse those rarely occuring semantic tags with more common ones.

4.6. Hyper-parameter optimization

As described in Chapter 3, training a neural network entails learning its weight and bias parameters with the support of a loss function. In the case of neural models implemented with our system, the loss function employed is the categorical cross-entropy loss. The only exception occurs when the last layer of the constructed network is a Conditional Random Field (CRF), in which case the loss function shown in Equation 3.15 is used.

One can define the categorical cross-entropy loss $\mathcal{H}$ as shown in Equation 4.1 by once again considering a reference training set $D = \{(W_i, T_i) : i = 1, \ldots, N\}$ where each pair (W_i, T_i) contains a sequence of tokens and its corresponding sequence of tags. We use the notation ϕ to denote the set of all network parameters, $\mathbf{T}_{ij}$ to represent the desired class for the j-th token of the i-th item in D as a one-hot encoded vector, and $\hat{\mathbf{T}}_{ij}$ to denote the probability vector predicted by a tagging model.

$$\mathcal{H}(\theta, D) = -\sum_{i=1}^{N} \sum_{j=1}^{|T_i|} \mathbf{T}_{ij} \log(\hat{\mathbf{T}}_{ij}) \tag{4.1}$$

However, neural networks do also contain other parameters that are set before the learning process begins and are not determined by training. These are denominated hyper-parameters, and our software offers the possibility of estimating them by using a brute-force exploration technique commonly referred to as grid search.

Thus, in order to choose the best possible set of hyper-parameters, we try to asses the suitability of user predefined hyper-parameter combinations using k-fold cross-validation. Our cross-validation approach first partitions the training data into $k = 3$ samples, and uses a single one of these samples for evaluating the performance of a network trained under a given combination of hyper-parameters on the other 2 samples. The cross-validation process is repeated 3 times with the sample used for evaluation changing each time, and the accuracy results on the different samples are averaged at the end. The hyper-parameter combination with the best performance is then selected.

Determining the parameter combinations to search is a yet another delicate issue. Since training neural network models is often costly both in terms of time and computational resources, it is infeasible to exhaustively explore the hyper-parameter combination space. One must then, choose which hyper-parameters would be more reasonable to optimize largely relying on experience. The results presented in this chapter correspond to models where we chose to optimize within the following hyper-parameters and values:

- Number of training epochs $\in \{10, 15\}$.
- Bach size $\in \{150, 200\}$.
- Dropout rate $\in \{0.1, 0.3\}$, which is applied to both the input and recurrent weights through the neural network [SHK+14].
- Number of hidden layers $\in \{1, 2\}$. This quantity refers to bidirectional layers in the case of bidirectional models.
- Number of hidden states $\in \{200, 300\}$.

Similarity, all models presented in this chapter invariably comply with the following fixed characteristics:

- Usage of the Adam algorithm for optimization [KB14].
- Usage of Rectified Linear Units (ReLU) as the default activation function in all layers except the output layer [NH10].
- Application of batch normalization following every layer.

Finally, we randomly explored hyper-parameter combinations and setups outside the values here described without much success. In particular, and as opposed to [BPB16], our software does not employ a residual bypass or an auxiliary loss function, and we found that these techniques results in worse or equal accuracy scores in our experiments. Increasing the number of hidden layers over 2 or the number of hidden states over 300 also does not seem to have a favorable impact in terms of results.

4.7. Evaluation

Having provided the necessary technical details, we will now proceed to present the results obtained after training and testing different models described in Chapter 3. While we naturally consider neural network taggers implemented using the software tool associated with the present thesis, we also employ taggers that rely on external implementations. In particular, we assess the performance of the *TnT* tagger [Bra00] in its default configuration and a CRF model trained using the *CRFsuite* software [Oka07]. In the latter case, the chosen architecture considers all possible 1-gram, 2-gram and 3-gram attributes within the window w_{i-2}^{i+2} at its i-th state.

This work explores neural network taggers in detail, and we evaluate unidirectional and bidirectional Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. We also consider coupling recurrent architectures with a CRF output layer as shown in Figure 3.5. The models for which we show results are all trained over 10 epochs using a batch size of 150, a single potentially bidirectional layer, and 300 hidden states. The dropout rate is set to 0.1 in LSTM-based models and to 0.3 in GRU-based models. Finally, these models use 300-dimensional *GloVe* embeddings trained on the Common Crawl corpus [PSM14] and 18-dimensional character embeddings initialized using the Glorot initialization method [GB10].

The tagging accuracy results of several systems used in our experiments are shown in Table 4.2. It can be seen that the Most Frequent Class (MFC) tagger delivers a reasonable performance on both the gold and silver test splits. However, the notable difference between the accuracy scores in those datasets is equally noticeable, with the silver test split showing nearly a 5% accuracy improvement compared to its gold counterpart. This suggests that despite the manual corrections, the silver test split still contains many mistakes favoring the commonly assigned semantic tags for each token. Thus, it must also be the case that a model performing remarkably well on the silver test split would not be so effective for universal semantic tagging in a real setting. From our perspective, we regard the gold test measure as being more reliable than any other for assessing the suitability of a tagger.

Sem-tagging performance					
Model	Features	Gold data		Silver data	
		Train	Test	Train	Test
MFC	w_1^n	84.138	**82.867**	89.118	87.523
TnT	w_1^n	90.633	**90.049**	94.624	93.459
CRFsuite	w_1^n	93.178	**89.126**	98.045	92.387
LSTM	$\vec{w}$	92.759	90.858	92.383	89.039
	$\vec{c}$	91.570	89.511	91.335	88.436
	$\vec{w} \oplus \vec{c}$	94.935	**91.591**	93.683	89.469
GRU	$\vec{w}$	90.777	90.565	90.348	88.997
	$\vec{c}$	89.371	89.774	88.969	87.601
	$\vec{w} \oplus \vec{c}$	91.947	**91.444**	91.644	89.833
BLSTM	$\vec{w}$	95.414	91.327	95.473	90.874
	$\vec{c}$	94.063	91.093	93.017	88.845
	$\vec{w} \oplus \vec{c}$	97.249	**92.118**	95.928	90.819
BGRU	$\vec{w}$	92.436	91.679	93.085	91.118
	$\vec{c}$	90.819	90.712	90.923	89.236
	$\vec{w} \oplus \vec{c}$	93.503	**92.441**	93.567	91.088
BLSTM-CRF	$\vec{w}$	95.715	92.148	95.460	91.031
	$\vec{c}$	94.376	91.650	93.413	89.055
	$\vec{w} \oplus \vec{c}$	97.089	**92.616**	95.984	90.767
BGRU-CRF	$\vec{w}$	91.078	90.009	91.120	89.312
	$\vec{c}$	90.420	90.214	91.524	89.787
	$\vec{w} \oplus \vec{c}$	93.529	**91.972**	93.822	91.234

Table 4.2: Accuracy scores obtained using the different models described in Chapter 3 trained jointly on the gold and silver training splits of the PMB. All numerical values represent the percentage of correctly predicted tags. The symbol $\vec{w}$ indicates the usage of word-level features and $\vec{c}$ denotes character-level features.

Both the *TnT* and the *CRFsuite* taggers are able to surpass the MFC baseline and obtain a very similar performance with approximately 90% accuracy on the gold test split. While this is not an entirely fair comparison since a major proportion of the silver data is actually tagged using *TnT* [ABE+17], it seems as if there is not really an advantage in using a more powerful CRF instead of an HMM tagger in the current state of affairs.

In turn, neural network taggers beat the non-neural models by a significant margin. Both LSTM and GRU models show that adding extra features and a bidirectional layer helps to obtain better scores. Furthermore, using a CRF in the output layer brings a clear improvement to the BLSTM model, but does not help the BGRU model that much.

All in all, our best performing tagger in terms of 3-fold cross-validation accuracy is the BLSTM-CRF model employing both word and character-level features, followed by the BGRU-CRF model using the same features. The earlier obtains an accuracy score of 92.6% on the gold test split, which represents an improvement of 8.7% with respect to the previous state-of-the-art model [BPB16]. Nonetheless, both the architectures used are similar in many ways, and the results are not directly comparable since [BPB16] uses different data and an older version of the Universal Semantic Tagset.

To conclude, we will look more in detail into the predictions being made by our models. To that effect, we consider the BLSTM-CRF model as shown in Table 4.2 and discuss its confusion matrix on the joint gold and silver test data. This is depicted in Figure 4.4, where the actual tags are shown in the vertical axis and the predicted tags in the horizontal axis. The matrix rows are independently normalized, meaning that the more intense the color of a semantic tag in the horizontal axis, the more often the colored semantic tag is assigned to the corresponding semantic tag in the vertical axis.

The semantic tags appearing in the confusion matrix are also ordered based on their frequency (Figure 4.3). Thus, the semantic tags appearing on top of the vertical axis and on the left of the horizontal axis are the most common. One can see that common semantic tags tend to be predicted perfectly and to not be confused with other tags. As the relative frequency of semantic tags grows lower, they are increasingly mistaken for more common semantic tags, which is an expected behavior given the nature of our data and tagger.

One can observe many low frequency `NAM` and `TIM` tokens being labeled as `CON` by our tagger. Similarly, many `ATT` semantic tags are also confused with `IST`, showing that our tagger is able to capture the coarse-grained information but struggles with making finer distinctions. Additionally, `EVE` tags seem particularly challenging and it is often the case that `EXG` and `EXT` are confused with `EXS`. Tokens with the semantic tag `GEO` are often mistakenly labeled `GPE` and contrariwise, and the tag `QUV` is also confused with `QUC`.

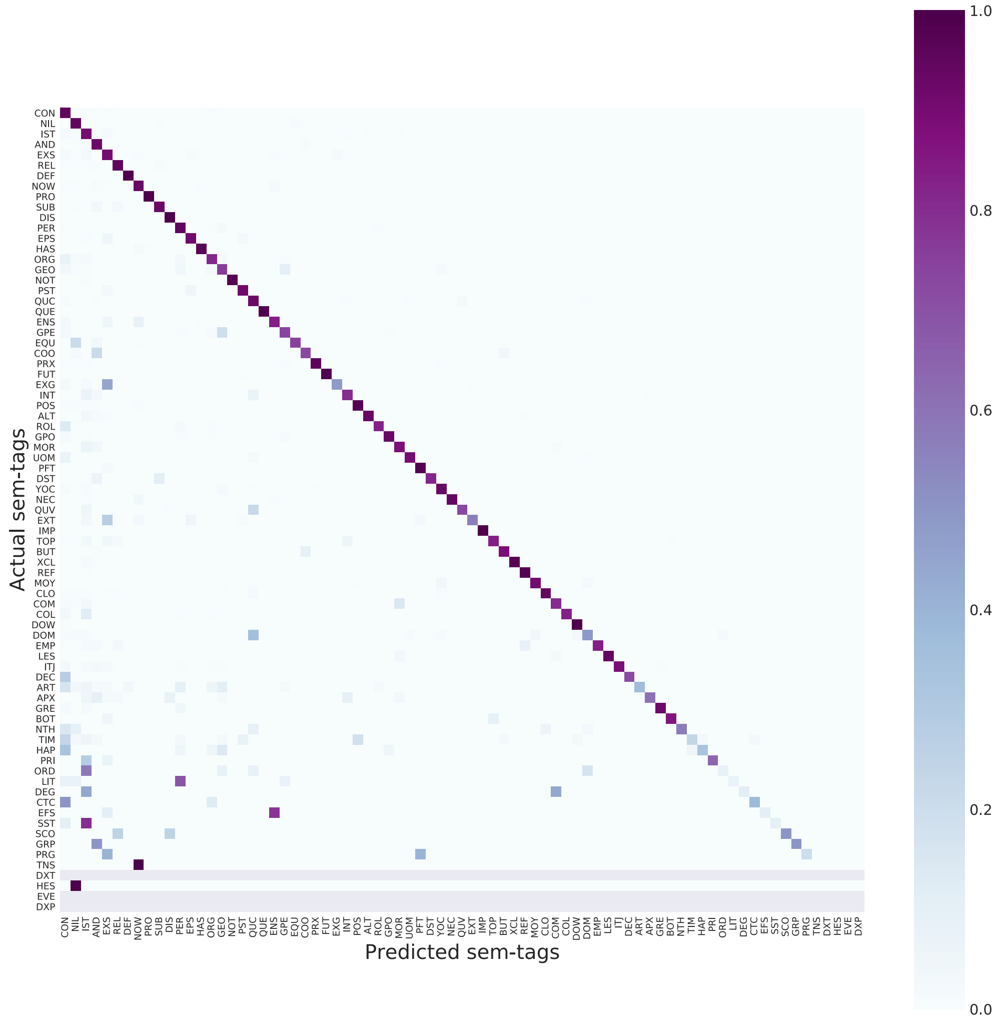

Figure 4.4: Confusion matrix produced by the BLSTM-CRF model in Table 4.2 when jointly predicting the semantic tags for the gold and silver test splits. Confusion frequencies are normalized by row, and semantic tags sorted according to they frequency. Grey regions indicate tags that were found in training but did not occur in the data considered.

However, these mistakes that our BLSTM-CRF tagger makes seem almost understandable if one considers the word embedding features employed, which give similar representations to tokens that have the same distributional properties. The conclusion that can be made is that word and character embeddings are not enough, and that we need other more specialized features which allow us to distinguish adjectival or eventual information in a fine-grained manner.

5. Semantic parsing and inference

Having described an implementation of a an accurate system for universal semantic tagging, we now wish to prove that the information conveyed by semantic tags can be useful in other natural language processing applications. We achieve this by using compositional semantic parsing together with semantics tags in a textual entailment task.

The principle of compositionality states that the meaning of a complex sentence is a function of the meaning of its parts [BB05]. Following this assumption, the compositional semantics approach which we consider in this thesis aims at assigning a semantic representation to each lexical item in a given phrase and, using the syntax of the language, combining those lexical semantics in order to create a complete meaning representation.

Such an approach requires 3 different components. First of all, one needs a formalism that can be used to represent meaning. A syntactic formalism able to construct a hierarchical structure that can guide the combination of meaning representations is also required [BB05]. Finally, one needs a computational model that defines the rules under which the very same meaning combinations can be performed. In the present thesis, we decided to employ λ-terms, Combinatory Categorial Grammars (CCG) [Ste96] and simply typed λ-calculus in order to achieve such goals.

5.1. Lambda calculus

The λ-calculus is one universal model of computation in which every expression is considered to be either a function or an variable, and where one can obtain reduced λ-terms by applying functions to arguments. Specifically, assuming a countably infinite set of variables $\mathcal{X}$, λ-terms can be constructed as follows:

$$\begin{aligned} &\textit{Variables:} && \text{If } x \in \mathcal{X}, \text{ then } x \text{ is a } \lambda\text{-term} \\ &\textit{Function application:} && \text{If } A \text{ and } B \text{ are } \lambda\text{-terms, then } A(B) \text{ is a } \lambda\text{-term} \\ &\textit{Function abstraction:} && \text{If } x \in \mathcal{X} \text{ and } A \text{ is a } \lambda\text{-term, then } (\lambda x.\, A) \text{ is a } \lambda\text{-term} \end{aligned} \tag{5.1}$$

Thus, the abstraction process allows us to bound a variable in a given formula by using the λ operator. These bounded variables can be understood as markers that explicitly indicate where we should substitute bits of information obtained during the course of semantic construction [BB05].

Furthermore, λ-terms can be reduced using the rules shown in Equation 5.2. The α-conversion rule allows to rename a bounded variable in the body of a λ-term in order to avoid name collisions, and the β-reduction rule replaces a bounded variable in the body of a λ-term with a provided argument.

$$\begin{aligned} \textit{α-conversion:} \quad & (\lambda x.\, A[x]) \rightarrow (\lambda z.\, A[z]) \\ \textit{β-reduction:} \quad & (\lambda x.\, A)(B) \rightarrow (A[B/x]) \end{aligned} \tag{5.2}$$

In the untyped version of λ-calculus, application is unconstrained, and this means that λ-terms can even be applied to themselves and create infinite loops. Such flexibility comes at the expense of increasing computational intractability. However, one can restrict application in simple typed λ-calculus by using typed λ-terms [Chu40]. In order to formally introduce the latter, we will first consider a set of possible types $\mathcal{Q}$ and the definition of pre-term in typed λ-calculus as outlined next:

$$\begin{aligned} \textit{Variables:} \quad & \text{If } x \in \mathcal{X}, \text{ then } x \text{ is a pre-term} \\ \textit{Function application:} \quad & \text{If } A \text{ and } B \text{ are pre-terms, then } A(B) \text{ is a pre-term} \\ \textit{Function abstraction:} \quad & \text{If } x \in \mathcal{X},\ A \text{ is a pre-term, and } T \in \mathcal{Q} \text{ is a type,} \\ & \quad \text{then } (\lambda x^T.\, A) \text{ is a pre-term} \end{aligned} \tag{5.3}$$

Pre-terms have the potential of becoming typed λ-terms after passing a set of typing judgments which are expressed as shown in Equation 5.4. In our formulation, we denote a partial function mapping variables to types as $\Gamma : \mathcal{X} \rightarrow \mathcal{Q}$, and use the symbols T and A to denote a particular type and pre-term respectively. In plain words, the left-hand side of a judgment declares variables with potential occurrence in A and their associated types, while the right hand side concludes that the type of the pre-term A must be T.

$$\Gamma : \mathcal{X} \rightarrow \mathcal{Q} \vdash A : T \tag{5.4}$$

Considering the basic form of a judgment, we specify the conditions under which types might be associated to pre-terms. These rules are commonly know as typing rules and are collected in Equation 5.5. Every well-typed λ-term can have its type automatically inferred by a type-checker via successive application of typing rules.

$$\begin{gathered}
\frac{i \in [1, n]}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i} \\[1em]
\frac{\Gamma \vdash A : T \rightarrow U \qquad \Gamma \vdash B : T}{\Gamma \vdash A(B) : U} \\[1em]
\frac{\Gamma, x : T \vdash A : U}{\Gamma \vdash \lambda x^T .\, A : T \rightarrow U}
\end{gathered} \tag{5.5}$$

5.2. Combinatory Categorial Grammars

Combinatory Categorial Grammars (CCG) define a syntactic formalism that accounts for many linguistic phenomena and captures the inherent connection between syntax and semantics [Ste96]. Typically, a CCG possesses a lexicon where each token is associated with a category containing syntactic and semantic information. In this thesis, we employ λ-terms as previously introduced in order to represent these categorical semantics.

In turn, we specify the categorical syntax of tokens using the leftmost notation, under which a rightward-combining functor over categories of syntactic type β into categories of syntactic type α is written as α/β, with the corresponding leftward combining functor over the same domain written as $\alpha\backslash\beta$. It follows that a transitive verb such as *borrow* might have a syntactic type $(S\backslash NP)/NP$, taking as arguments an object and a subject in the form of Noun Phrases (NP) and heading a Sentence (S) category.

CCGs use a set of combinators that receive either 1 or 2 categories in order to combine them together into a larger category [Ste96]. Because these combinators apply simultaneously on both the syntactic and the semantic parts of each category, they can be used to construct larger blocks of syntactic and semantic structure. The most basic combination rules are the functional application rules, defined as follows:

$$\begin{aligned} A/B : f \quad B : g \ &\rightarrow A : f(g) \\ B : g \quad A\backslash B : f \ &\rightarrow A : f(g) \end{aligned} \tag{5.6}$$

The rules in Equation 5.6 are also known as forward application (>) and backward application (<). They state that a category with syntactic type A/B can be combined with a category of syntactic type B on its right or left to produce a category of syntactic type A. The new semantics of this combination can be derived by applying f to g.

$$\begin{array}{cccc}
\text{Several} & \text{employees} & \text{drink} & \text{coffee} \\
\hline
NP/N & N & (S\backslash NP)/NP & NP \\
\lambda F \lambda G.\, \exists x(F(x) \wedge G(x)) & \lambda x.\, \mathsf{employee}(x) & \lambda Q_1 \lambda Q_2.\, Q_2(\lambda x.\, Q_1(\lambda y.\, \mathsf{drink}(x,y))) & \lambda F.\, \exists y(\mathsf{coffee}(y) \wedge F(y)) \\
\hline
\multicolumn{2}{r}{>} & \multicolumn{2}{r}{>} \\
\multicolumn{2}{c}{NP} & \multicolumn{2}{c}{S\backslash NP} \\
\multicolumn{2}{c}{\lambda G.\, \exists x(\mathsf{employee}(x) \wedge G(x))} & \multicolumn{2}{c}{\lambda Q_2.\, Q_2(\lambda x.\, \exists y(\mathsf{coffee}(y) \wedge \mathsf{drink}(x,y)))} \\
\hline
\multicolumn{4}{r}{<} \\
\multicolumn{4}{c}{S} \\
\multicolumn{4}{c}{\exists x(\mathsf{employee}(x) \wedge \exists y(\mathsf{coffee}(y) \wedge \mathsf{drink}(x,y)))}
\end{array}$$

Figure 5.1: Simplified CCG derivation tree with its syntax and semantics corresponding to the sentence *Several employees drink cofee*, inspired from an example in [MGMMB16].

.

Nonetheless, there is a number of linguistic phenomena which cannot be modeled by the functional application rules. In order to gain more expressive power, we require another pair of rules for functional composition. Forward composition (>**B**) and backward composition (<**B**) rules are defined in the same fashion as function application:

$$\begin{aligned} A/B : f \quad B/C : g \ &\rightarrow A/C : \lambda x.f(g(x)) \\ B\backslash C : g \quad A\backslash B : f \ &\rightarrow A\backslash C : \lambda x.f(g(x)) \end{aligned} \tag{5.7}$$

The next combination rules which we will consider are forward type raising (>**T**) and backward type raising (<**T**), which are unary rules that allow to cast simple categories into functions. These functions expect a third function that takes the original category as its argument [JM09].

$$
\begin{aligned}
X : a &\rightarrow T/(T\backslash X) : \lambda f.\, f(a) \\
X : a &\rightarrow T\backslash(T/X) : \lambda f.\, f(a)
\end{aligned}
\tag{5.8}
$$

Finally, we can add yet another additional rule in order to address the syntactic phenomenon of coordination, where two categories of the same type are joined together and form a larger category, as it happens with the conjunctions *and* or *or*. The definition of the coordination rule in its syntactic part is shown in Equation 5.9.

$$X \text{ conj } X \rightarrow X \tag{5.9}$$

Additional combination rules have been suggested for CCGs in recent years, attempting to include as much lexical information as possible in the lexicon [KKS15]. However, the current thesis omits them in favor of succinctness.

5.2.1. Learning optimal grammars

Using CCGs to find the correct parse tree for a given sentence is a challenging problem, since sentences and their corresponding logical forms can be potentially derived from a large number of parse trees. In order to formalize the problem and discuss its possible solutions, we will consider a probabilistic generalization of CCGs known as Probabilistic Combinatory Categorial Grammar (PCCG) [ZC05].

The PCCG formalism defines the conditional probability $P(L, T \mid S)$ over pairs (L, T) of possible semantic representations and parse trees for a given sentence S. Assuming a function $\mathbf{f}(L, T, S) \in \mathbb{R}^d$ which maps triples to d-dimensional feature vectors and a set of model parameters $\theta \in \mathbb{R}^d$, one can define the probability of a pair (L, S) in the log-linear model as follows:

$$P(L, T \mid S; \theta) = \frac{e^{\mathbf{f}(L,T,S)\cdot\theta}}{\sum\limits_{(L,T)} e^{\mathbf{f}(L,T,S)\cdot\theta}} \tag{5.10}$$

Next, after marginalizing out T by summing over all possible parse trees, we can express the most probable semantic representation $\hat{L}$ as shown in Equation 5.11. The solution $\hat{L}$ can be found by using a combination of beam search and parsing algorithms relying on dynamic programming such as the CYK algorithm, which is conceptually similar to the Viterbi algorithm described in Chapter 3.

$$\hat{L} = \arg\max_{L} P(L \mid S; \theta) = \arg\max_{L} \sum_{T} P(L, T \mid S; \theta) \tag{5.11}$$

Training a PCCG model of this form involves learning the parameters θ and perhaps the lexicon. Given a dataset $D = \{(S_i, L_i) : i = 1, \ldots, N\}$, we can define the log-likelihood function shown in Equation 5.12 and find the optimal parameter values by using gradient descent methods similar to the ones discussed in Chapter 3. The lexicon can also be learned by re-estimating θ while successively adding lexical entries [ZC07], or otherwise extracted from annotated data such as the CCGbank [HS07].

$$\mathcal{O}(\theta, D) = \sum_{i=1}^{N} \log P(L_i \mid S_i; \theta) = \sum_{i=1}^{N} \log \left(\sum_{T} P(L_i, T \mid S_i; \theta) \right) \tag{5.12}$$

5.3. Parsing systems (ccg2lambda)

The *ccg2lambda* system is a software tool that can construct higher-order logical representations of sentences by employing λ-calculus and CCG parse trees [MGMMB16] in the manner which we described in the current chapter. The system employs popular CCG parsing tools such as *C&C* [CC07] in order to automatically and reliably construct such parse trees.

However, *ccg2lambda* relies on manual specification of lexical semantics and on precise indications on how to construct meaning representations. The hypothesis which we adopt in this thesis is that the specification process could be made simpler by considering the information provided by semantic tags.

5.3.1. Semantic templates

Linguistically motivated and manually defined semantic templates are employed in order to define the semantics corresponding to CCG parse trees within *ccg2lambda* [MMMB]. Each one of the rules contained in a semantic template needs to specify a valid CCG syntactic type, which serves to identify lexical or internal nodes from CCG derivation trees in which the rule applies. Similarly, each rule needs to provide the corresponding semantics at the node of application in the form of a λ-term.

Needless to say, such a manual specification procedure is labor-intensive. Furthermore, it suffers from the same problem which motivated the introduction of semantic tags in Section 2. We will once more employ determiners in order to illustrate the inconvenience of semantic templates by considering the 2 semantic template rules shown next:

$$\begin{aligned}
&\textit{Syntactic type:} && NP[nb]/N \\
&\textit{Semantic type:} && \lambda E \lambda F_1 \lambda F_2 \lambda F_3 .\, \forall x (F_1(x) \to (F_2(x) \to \neg F_3(x))) \\
&\textit{Surface form:} && \text{'}\textit{no}\text{'} \\
\\
&\textit{Syntactic type:} && NP[nb]/N \\
&\textit{Semantic type:} && \lambda E \lambda F_1 \lambda F_2 \lambda F_3 .\, \exists x (F_1(x) \wedge F_2(x) \wedge F_3(x))) \\
&\textit{Surface form:} && \text{'}\textit{the}\text{'}
\end{aligned} \tag{5.13}$$

Both the rules in Equation 5.13 specify different semantics that a lexical node with syntactic type $NP[nb]/N$ in the CCG derivation can claim as its own. While the syntactic type is the same on both rules, the lexical semantics that the tokens *no* and *the* convey are the exact opposite of each other.

Semantic template rules have the option of restricting their application by using different kinds of information, such as Part-Of-Speech (POS) tags. However, in Equation 5.13 both the tokens considered are mapped to the same POS tag `DT`, and thus one must list the lexical form of the tokens in order to force enforce a differentiating restriction.

5.3.2. Semantically tagged templates

The information provided by semantic tags can help us simplify the process of writing semantic templates. Not only semantic tags group together tokens that have similar meanings, but they also provide fine-grained semantic distinctions, as discussed in Chapter 2. We will now consider the semantic template rules shown in Equation 5.13 and rewrite them in Equation 5.14, this time using semantic tags for restricting rule application.

$$\begin{array}{rl}
\textit{Syntactic type:} & NP[nb]/N \\
\textit{Semantic type:} & \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \forall x (F_1(x) \rightarrow (F_2(x) \rightarrow \neg F_3(x))) \\
\textit{Semantic tag:} & \texttt{NOT} \\
\\
\textit{Syntactic type:} & NP[nb]/N \\
\textit{Semantic type:} & \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \exists x (F_1(x) \wedge F_2(x) \wedge F_3(x))) \\
\textit{Semantic tag:} & \texttt{DEF}
\end{array} \tag{5.14}$$

Rules which feature semantic tags allow us to subsume many other rules that apply to other surface forms mapped to the same tag. For example, there is no further need to specify a rule such as the one listed in Equation 5.15, since the token *neither* is already associated to the semantic tag `NOT` in the given context and covered by the the first rule in Equation 5.14. In the original templates, one must nonetheless list all such words, which is not only a linguistic effort but also a procedure prone to errors by omission.

Appendix A gives an exhaustive list of template rules based on the work of [MMMB] which we created or modified from standard templates in order to introduce semantic tags. In doing so, we were able to build a semantically tagged template containing 114 rules with more expressive power than a standard template containing 137 rules.

$$\begin{array}{rl} \textit{Syntactic type:} & NP[nb]/N \\ \textit{Semantic type:} & \lambda E \lambda F_1 \lambda F_2 \lambda F_3 . \forall x(F_1(x) \rightarrow (F_2(x) \rightarrow \neg F_3(x))) \\ \textit{Surface form:} & \text{'}\textit{neither}\text{'}(\mapsto \boxed{\texttt{NOT}}) \end{array} \tag{5.15}$$

5.4. Recognizing Textual Entailment

An hypothesis which we entertain in the current thesis is that semantic tags can prove useful when integrated to other systems not purely concerned with semantic parsing in an end-to-end manner. Attempting to do exactly that also serves as a means to empirically evaluate the usefulness of semantic tags. In this section we will consider the task of Recognizing Textual Entailment (RTE) for such a purpose.

Textual entailment can be defined as a relationship between a given text T and a hypothesis H. We will claim that T entails H when humans are likely to judge that the meaning of H directly follows from the meaning of T. Thus, the RTE task requires systems to decide automatically between 3 different verdicts for entailment pairs (T, H), either stating that the entailment relation is present, is not present, or that a decision cannot be made based on the facts appearing in T alone.

Successful approaches to RTE often use symbolic representations of the sentences both in the text and the hypothesis [Abz15]. In particular, a system can be constructed by using the higher-order logical representations produced by the *ccg2lambda* system [MGMMB16]. To do so, one can construct a theorem of the form shown in Equation 5.16, where we assume the existence of a set of premises $p_1, \ldots, p_n$ and a single conclusion c without loss of generality.

$$p_1 \rightarrow \cdots \rightarrow p_n \rightarrow c \tag{5.16}$$

The next step is to assign semantic types to the terms appearing within our representations, which we will accomplish by considering the two basic types E for entities and Prop for propositions. Specific higher-order denotations described in [MMMB] and not replicated here are assigned pre-defined types. For example, the predicates manage and believe can be mapped to the type $\mathsf{Prop} \to \mathsf{E} \to \mathsf{Prop}$.

Finally, we present our typed theorem to the *Coq* higher-order prover [BC10], which automatically determines whether the entailment relationship exists (a *yes* verdict), whether the entailment of the negation exists (a *no* verdict), or does not (an *unknown* verdict).

5.4.1. The FraCas Test Suite

The FraCas Test Suite contains a set of 346 inference problems which are designed to specify the aspects of language that natural language inference systems should be able to capture [CCE+96]. Each example contains 1 or 2 premises, a question, a hypothesis, and a judgement of the relation between the premise and the hypothesis. A summary of the dataset can be seen in Table 5.1.

Topic	Total count	Percentage	Single-premise
Quantifiers	80	23%	50
Plurals	33	10%	24
Anaphora	28	8%	6
Ellipsis	55	16%	25
Adjectives	23	7%	15
Comparatives	31	9%	16
Temporal	75	22%	39
Verbs	8	2%	8
Attitudes	13	4%	9

Table 5.1: Distribution of inference problems in the FraCas Test Suite according to their topic. For each topic we show the number of problems, the percentage it represents with respect to the entire dataset, and the number of associated problems with a single premise.

We will shown next an example of a problem containing multiple premises and a *yes* verdict. As explained previously, these problem premises and verdicts can be transformed into typed higher-order logical expressions and be automatically verified for entailment.

`fracas-027  answer:  yes`

$$\begin{array}{ll} p_1: & \text{A few committee members are from Sweden.} \\ p_2: & \text{All committee members are people.} \\ p_3: & \text{All people who are from Sweden are from Scandinavia.} \\ c: & \text{At least a few committee members are from Scandinavia.} \end{array} \tag{5.17}$$

5.4.2. Performance comparison

The suggested system for the RTE task based on *ccg2lambda* and *Coq* [MGMMB16] is evaluated on the FraCas dataset both by using a standard semantic template and its corresponding semantically tagged translation suggested by us and given in Appendix A. The results are grouped by relevant topics on Table 5.2, where we measure the accuracy on the entailment decisions *yes*, *no*, and *unknown*.

Topic	**Standard template**	**Sem-tag template**
Quantifiers	78%	76%
Plurals	67%	67%
Adjectives	68%	68%
Comparatives	48%	48%
Verbs	62%	62%
Attitudes	77%	77%
Total	**69%**	**68%**

Table 5.2: Performance comparison between a standard semantic template [MGMMB16] and its semantically tagged equivalent suggested in this thesis.

We can observe that the performance of our template is very close to that of the original template, which represents the current state-of-the-art [MGMMB16]. In fact, despite our template having less rules, the only noticeable difference is the non-existential interpretation of the quantifier *few* in the original template versus the existential interpretation given by the semantically tagged template. We consider that this is a positive result that shows the potential future capabilities of semantic tag-based approaches in natural language processing applications.

Finally, it is worth noting that many of the mistakes that our modified system suffers from are caused by either tokens being assigned incorrect semantic tags or incorrect CCG syntactic types. The fact that we are employing 2 statistical models as preprocessing steps prior to semantic parsing increases the risk of being exposed to errors in a given sentence. Thus, it might be advantageous to consider approaches with the least error rate per sentence instead of per word.

6. Conclusions and future work

In this work we analyzed the topic of universal semantic tagging [AB17], on which we conducted an intrinsic and an extrinsic evaluation. On the one hand, we showed that it is possible to use state-of-the-art techniques in sequence tagging in order to design reliable models capable of assigning semantic tags to word tokens. On the other hand, we were able to employ the implemented models and the expressiveness of semantic tags jointly with other systems, creating a pipeline able to effectively solve a separate task.

Thus, the first half of this thesis is concerned with building a software tool implementing various tagging models[1] using the data from the Parallel Meaning Bank (PMB) [ABE+17]. Our best performing model is a bidirectional Long Short-Term Memory (LSTM) neural architecture [HS97] coupled with a Conditional Random Field (CRF) [LMP01] that uses both word-level and character-level features. Compared to other neural semantic taggers like [BPB16], this model obtains competitive results[2]. We believe that our work constitutes a useful tool providing a solid baseline for future research.

On the second half of this thesis we analyze the usefulness of semantic tags from a practical perspective by applying them to the Recognizing Textual Entailment (RTE) problem. Using simply typed λ-calculus as a computational model, we were able to write semantic templates that define how to build higher-order logical representations from syntactic derivation trees produced by Combinatory Categorial Grammars (CCG) [Ste96]. Because our semantic templates employ semantic tags, they present a wider coverage than previous attempts while also containing fewer rules [MMMB].

Finally, we integrated our novel semantic templates within the *ccg2lambda* system in order to perform compositional semantic parsing of natural language sentences into the defined logical representations [MGMMB16]. By also employing the *Coq* theorem prover in order to determine whether entailment relations between sets of these logical representations exist [BC10], we were able to obtain state-of-the-art results in the RTE task associated to the FraCaS dataset [CCE+96].

[1]Our implementation is available from `https://github.com/ginesam/semtagger`.

[2]The results obtained from our experiments are not comparable to those in [BPB16], since the training/test data and the version of the semantic tagset used differ.

Notwithstanding the exploratory nature of the work here presented, we believe that ours is a contribution that shows the real possibility of starting to introduce universal semantic tagging as an auxiliary task in natural language processing applications. In the future, we would like to continue exploring in the direction of RTE by defining semantic templates using neo-Davidsonian event semantics [Dav67] or dynamic semantics [Kam81]. Similarly, we would like to measure the performancce of our modified system in datasets other than FraCaS which contain more lexical variability, such as the SICK dataset [MMB+14].

From where we stand, it is also expected that the work on universal semantic tagging and the PMB will continue in the upcoming years, providing more data that will allow to train more precise semantic taggers, and updating the dataset in order to increase coverage and to simplify the learning process, while keeping in mind the suitability of semantic tags for cross-lingual applications. In turn, we hope that these developments bring together improvements to other areas such as natural language question answering, entity linking or even machine translation.

Bibliography

[AB17] Lasha Abzianidze and Johan Bos. Towards Universal Semantic Tagging. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS 2017)*, pages 1–6, 2017.

[ABE$^+$17] Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247. Association for Computational Linguistics, 2017.

[Abz15] Lasha Abzianidze. A Tableau Prover for Natural Logic and Language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502. Association for Computational Linguistics, 2015.

[BB05] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI Publications, 2005.

[BC10] Yves Bertot and Pierre Castran. *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[BCS$^+$04] Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. Wide-coverage Semantic Representations from a CCG Parser. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04. Association for Computational Linguistics, 2004.

[Bo08] Egwin Boschman and Hendrikus J.A. op den Akker. A Neural Network Based Dutch Part of Speech Tagger. In *Proceedings of the twentieth Belgian-Dutch Artificial Intelligence Conference (BNAIC 2008)*, pages 217–224. Twente University Press (TUP), 2008.

[BPB16] Johannes Bjerva, Barbara Plank, and Johan Bos. Semantic Tagging with Deep Residual Networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 3531–3541. Association for Computational Linguistics (ACL), 2016.

[Bra00] Thorsten Brants. TnT: A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, ANLC '00, pages 224–231. Association for Computational Linguistics, 2000.

[Bri95] Eric Brill. Transformation-based Error-driven Learning and Natural Language Processing: A Case Study in Part-of-speech Tagging. *Comput. Linguist.*, 21(4):543–565, 1995.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[C$^+$15] François Chollet et al. Keras. https://keras.io, 2015.

[CC07] Stephen Clark and James R. Curran. Wide-coverage Efficient Statistical Parsing with CCG and Log-linear Models. *Comput. Linguist.*, 33(4):493–552, 2007.

[CCE$^+$96] Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Josef Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman, Ted Briscoe, Holger Maier, and Karsten Konrad. Using the Framework, 1996.

[Chu40] Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.

[CKPS92] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC '92, pages 133–140. Association for Computational Linguistics, 1992.

[CvMG$^+$14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

[CZH$^+$17] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 77–87. Curran Associates, Inc., 2017.

[Dav67] Donald Davidson. The Logical Form of Action Sentences. In *The Logic of Decision and Action*. University of Pittsburgh Press, 1967.

[DSZ14] Cícero Nogueira Dos Santos and Bianca Zadrozny. Learning Character-level Representations for Part-of-speech Tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1818–II–1826, 2014.

[FK79] W. Nelson Francis and Henry Kucera. The Brown Corpus: A Standard Corpus of Present-Day Edited American English, 1979. Brown University Liguistics Department.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

[GRDB07] Samuel S. Gross, Olga Russakovsky, Chuong B. Do, and Serafim Batzoglou. Training Conditional Random Fields for Maximum Labelwise Accuracy. In *Advances in Neural Information Processing Systems 19*, pages 529–536. MIT Press, 2007.

[Gur97] Kevin Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA, 1997.

[Hay98] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, 2nd edition, 1998.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[HS07] Julia Hockenmaier and Mark Steedman. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Comput. Linguist.*, 33(3):355–396, 2007.

[HSW89] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.*, 2(5):359–366, 1989.

[HXY15] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, abs/1508.01991, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[ID10] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing.* Chapman & Hall, 2nd edition, 2010.

[Jan04] Artur Janicki. Application of Neural Networks for POS Tagging and Intonation Control in Speech Synthesis for Polish. In *Soft Computing and Intelligent Systems (SCIS 2004)*, 2004.

[JM09] Daniel Jurafsky and James H. Martin. *Speech and Language Processing.* Prentice-Hall, Inc., 2nd edition, 2009.

[Kam81] Hans Kamp. A Theory of Truth and Semantic Representation. In *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum, 1981.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[KJSR16] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. Character-Aware Neural Language Models. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 2741–2749. AAAI press, 2016.

[KKS15] Marco Kuhlmann, Alexander Koller, and Giorgio Satta. Lexicalization and Generative Power in CCG. *Comput. Linguist.*, 41(2):215–247, 2015.

[Lip15] Zachary Chase Lipton. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, abs/1506.00019, 2015.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., 2001.

[Man11] Christopher D. Manning. Part-of-speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing '11, pages 171–189. Springer-Verlag, 2011.

[MCCD13] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781, 2013.

[MGMMB16] Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. ccg2lambda: A Compositional Semantics System. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90. Association for Computational Linguistics, 2016.

[MH16] Xuezhe Ma and Eduard Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics, 2016.

[MMB$^+$14] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014.

[MMMB] Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015.*

[MMS93] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330, 1993.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814. Omnipress, 2010.

[Oka07] Naoaki Okazaki. CRFsuite: A fast implementation of Conditional Random Fields (CRFs), 2007. URL: `http://www.chokkan.org/software/crfsuite/`.

[PGCB13] Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. How to Construct Deep Recurrent Neural Networks. *CoRR*, abs/1312.6026, 2013.

[PSG16] Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418. Association for Computational Linguistics, 2016.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014.

[Sch94] Helmut Schmid. Part-of-speech Tagging with Neural Networks. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 172–176, 1994.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SP97] Mike Schuster and Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.*, 45(11):2673–2681, 1997.

[SRLK14] Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. Part-of-Speech Tagging using Conditional Random Fields: Exploiting Sub-Label Dependencies for Improved Accuracy. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 259–264. Association for Computational Linguistics, 2014.

[Ste96] Mark Steedman. *Surface Structure and Interpretation.* Linguistic inquiry monographs. MIT Press, 1996.

[ZC05] Luke S. Zettlemoyer and Michael Collins. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, pages 658–666. AUAI Press, 2005.

[ZC07] Luke Zettlemoyer and Michael Collins. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[ZSV14] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. *CoRR*, abs/1409.2329, 2014.

A. Semantically tagged template rules

Here we list a collection of rules appearing in the semantic templates used to perform the entailment experiments described in Chapter 5. These rules are based on those already being used in the *ccg2lambda* system [MGMMB16, MMMB], with occasional variations and the incorporation of semantic tags. Note that the list here is not exhaustive, since we only report the rules that contribute to defining the meaning of semantic tags.

Semantic tags in this appendix group nodes from Combinatory Categorial Grammar (CCG) [Ste96] derivation trees where we consider that they can be used to further advance in the construction of a meaning representation. The notations $_{(1)}$, $_{(2)}$ and $_{(*)}$ respectively indicate that the corresponding semantic tag must be associated to the left child, the right child or any children of the given node in order for the rule semantics to apply. Features appearing within syntactic types follow the ones used in the CCGbank [HS07].

$$\texttt{DEF} = \left\{ \; _{(1)}NP[nb]/N \quad \mapsto \quad \lambda E \lambda F_1 \lambda F_2 \lambda F_3 .\, \exists x (F_1(x) \wedge F_2(x) \wedge F_3(x)) \; \right\}$$

$$\texttt{HAS} = \left\{ \begin{array}{lll} _{(1)}(NP[nb]/N)\backslash NP & \mapsto & \lambda E \lambda Q \lambda F_1 \lambda F_2 \lambda F_3 .\, \exists x ((Q(\lambda w.\, \top, \lambda y.\, \mathbf{rel}(x, y)) \wedge F_1(x)) \\ & & \quad \wedge F_2(x) \wedge F_3(x)) \end{array} \right\}$$

$$\texttt{IST} = \left\{ \; _{(1)}S[adj]\backslash NP \quad \mapsto \quad \lambda E \lambda Q .\, Q(\lambda w.\, \top, \lambda x.\, E(x)) \; \right\}$$

$$\texttt{SST} = \left\{ \; _{(1)}S[adj]\backslash NP \quad \mapsto \quad \lambda E \lambda Q .\, Q(\lambda w.\, \top, \lambda x.\, E(x)) \; \right\}$$

$$\texttt{REL} = \left\{ \; _{(1)}((S\backslash NP)\backslash(S\backslash NP))/S[dcl] \quad \mapsto \quad \lambda E \lambda S \lambda V \lambda Q .\, E(S, V(Q)) \; \right\}$$

$$\texttt{EQU} = \left\{ \; _{(1)}NP\backslash NP \quad \mapsto \quad \lambda L \lambda Q_1 \lambda Q_2 \lambda F_1 \lambda F_2 .\, (Q_2(F_1, F_2) \wedge Q_1(F_1, F_2)) \; \right\}$$

$$\texttt{UNE} = \left\{ \begin{array}{rcl} {}_{(1)}N & \mapsto & \lambda E.\, E \end{array} \right\}$$

$$\texttt{GRP} = \left\{ \begin{array}{rcl} {}_{(1)}NP\backslash NP & \mapsto & \lambda L \lambda Q_1 \lambda Q_2 \lambda F_1 \lambda F_2.\, (Q_2(F_1, F_2) \wedge Q_1(F_1, F_2)) \end{array} \right\}$$

$$\texttt{LOG} = \left\{ \begin{array}{rcl} {}_{(1)}N/N & \mapsto & \lambda E \lambda X.\, X \end{array} \right\}$$

$$\texttt{ALT} = \left\{ \begin{array}{rcl} {}_{(1)}(S\backslash NP)/(S\backslash NP) & \mapsto & \lambda E \lambda V \lambda Q.\, V(Q) \end{array} \right\}$$

$$\texttt{NIL} = \left\{ \begin{array}{rcl} {}_{(1)}S & \mapsto & \lambda L \lambda S.\, S \\ {}_{(1)}NP & \mapsto & \lambda L \lambda R.\, L \\ {}_{(1)}S\backslash NP & \mapsto & \lambda L \lambda R.\, L \\ {}_{(1)}NP\backslash NP & \mapsto & \lambda L \lambda Q_1 \lambda Q_2 \lambda F_1 \lambda F_2.\, (Q_2(F_1, F_2) \vee Q_1(F_1, F_2)) \\ {}_{(1)}S\backslash S & \mapsto & \lambda L \lambda S.\, S \\ {}_{(1)}. & \mapsto & \lambda S \lambda X.\, X \end{array} \right\}$$

$$\texttt{DIS} = \left\{ \begin{array}{rcl} {}_{(1)}S\backslash S & \mapsto & \lambda L \lambda S_1 \lambda S_2.\, (S_1 \wedge S_2) \\ {}_{(1)}NP\backslash NP & \mapsto & \lambda L \lambda Q_1 \lambda Q_2 \lambda F_1 \lambda F_2.\, (Q_2(F_1, F_2) \vee Q_1(F_1, F_2)) \\ {}_{(1)}N/N & \mapsto & \lambda L \lambda F_1 \lambda F_2 \lambda x.\, (F_1(x) \vee F_2(x)) \\ {}_{(1)}N\backslash N & \mapsto & \lambda L \lambda F_1 \lambda F_2 \lambda x.\, (F_1(x) \vee F_2(x)) \\ {}_{(1)}(N/N)/(N/N) & \mapsto & \lambda L \lambda M_1 \lambda M_2 \lambda F \lambda x.\, M_1(M_2(F), x) \\ {}_{(1)}(N/N)\backslash(N/N) & \mapsto & \lambda L \lambda M_1 \lambda M_2 \lambda F \lambda x.\, (M_1(F, x) \vee M_2(F, x)) \\ {}_{(1)}NP[nb]/N & \mapsto & \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\, \exists x (F_1(x) \wedge F_2(x) \wedge F_3(x)) \\ {}_{(1)}(S\backslash NP)\backslash(S\backslash NP) & \mapsto & \lambda L \lambda V_1 \lambda V_2 \lambda Q.\, Q(\lambda w.\, \top, \lambda x.\, (V_1(\lambda F_1 \lambda F_2.\, F_2(x)) \\ & & \quad \vee V_2(\lambda F_1 \lambda F_2.\, F_2(x)))) \end{array} \right\}$$

$$\texttt{IMP} = \left\{ \begin{array}{rcl} {}_{(1)}(S/S)/S[dcl] & \mapsto & \lambda E \lambda S_1 \lambda S_2.\, (S_1 \rightarrow S_2) \end{array} \right\}$$

$$\texttt{AND} = \left\{ \begin{array}{rcl} {}_{(1)}S\backslash S & \mapsto & \lambda L \lambda S_1 \lambda S_2.\,(S_1 \wedge S_2) \\ {}_{(1)}NP\backslash NP & \mapsto & \lambda L \lambda Q_1 \lambda Q_2 \lambda F_1 \lambda F_2.\,(Q_2(F_1,F_2) \wedge Q_1(F_1,F_2)) \\ {}_{(1)}N\backslash N & \mapsto & \lambda L \lambda F_1 \lambda F_2 \lambda x.\,(F_1(x) \wedge F_2(x)) \\ {}_{(1)}(N/N)\backslash(N/N) & \mapsto & \lambda L \lambda M_1 \lambda M_2 \lambda F \lambda x.\,M_1(M_2(F),x) \\ {}_{(1)}(N/N)/(N/N) & \mapsto & \lambda L \lambda M_1 \lambda M_2 \lambda F \lambda x.\,M_1(M_2(F),x) \\ {}_{(1)}NP & \mapsto & \lambda E \lambda F_1 \lambda F_2.\,\forall x(F_1(x) \rightarrow F_2(x)) \\ {}_{(1)}NP[nb]/N & \mapsto & \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\,\forall x(F_1(x) \rightarrow (F_2(x) \rightarrow F_3(x))) \\ {}_{(1)}NP/NP & \mapsto & \lambda E \lambda Q \lambda F_1 \lambda F_2.\,\forall x(Q(\lambda w.\top, \lambda y.\,((x=y) \\ & & \quad \wedge F_1(y))) \rightarrow F_2(x)) \\ {}_{(1)}NP\backslash NP & \mapsto & \lambda E \lambda Q \lambda F_1 \lambda F_2.\,\forall x(Q(\lambda w.\top, \lambda y.\,((x=y) \\ & & \quad \wedge F_1(y))) \rightarrow F_2(x)) \\ {}_{(1)}(S\backslash NP)/(S\backslash NP) & \mapsto & \lambda E \lambda V \lambda Q.\,\forall x(Q(\lambda w.\top, \lambda y.\,(x=y)) \rightarrow V(\lambda F_1 \lambda F_2.\,F_2(x))) \\ {}_{(1)}(S\backslash NP)\backslash(S\backslash NP) & \mapsto & \lambda E \lambda V \lambda Q.\,\forall x(Q(\lambda w.\top, \lambda y.\,(x=y)) \rightarrow V(\lambda F_1 \lambda F_2.\,F_2(x))) \end{array} \right\}$$

$$\texttt{NOT} = \left\{ \begin{array}{rcl} {}_{(*)}NP & \mapsto & \lambda C \lambda F_1 \lambda F_2.\,\neg\exists x(C(x) \wedge F_1(x) \wedge F_2(x)) \\ {}_{(1)}NP[nb]/N & \mapsto & \lambda E \lambda F_1 \lambda F_2 \lambda F_3.\,\forall x(F_1(x) \rightarrow (F_2(x) \rightarrow \\ & & \quad \neg F_3(x))) \\ {}_{(1)}((S\backslash NP)\backslash(S\backslash NP))/(S[ng]\backslash NP) & \mapsto & \lambda E \lambda V_1 \lambda V_2 \lambda Q.\,Q(\lambda w.\top, \lambda x.\,V_1(\lambda F_1 \lambda F_2. \\ & & \quad (V_2(\lambda G_1 \lambda G_2.\,G_2(x)) \wedge \neg F_2(x)))) \\ {}_{(1)}(S\backslash NP)\backslash(S\backslash NP) & \mapsto & \lambda E \lambda V \lambda Q.\,Q(\lambda w.\top, \lambda x.\,\neg V(\lambda F_1 \lambda F_2.\,F_2(x))) \\ {}_{(1)}(S[adj]\backslash NP)/(S[adj]\backslash NP) & \mapsto & \lambda E \lambda V \lambda Q.\,Q(\lambda w.\top, \lambda x.\,\neg V(\lambda F_1 \lambda F_2.\,F_2(x))) \end{array} \right\}$$

$$\texttt{DSC} = \left\{ \begin{array}{rcl} {}_{(1)}(S/S)/S[dcl] & \mapsto & \lambda E \lambda S_1 \lambda S_2.\,(S_1 \wedge S_2) \end{array} \right\}$$

$$\texttt{COO} = \left\{ \begin{array}{rcl} {}_{(1)}(S\backslash NP)\backslash(S\backslash NP) & \mapsto & \lambda L \lambda V_1 \lambda V_2 \lambda Q.\,Q(\lambda w.\top, \lambda x.\,(V_1(\lambda F_1 \lambda F_2.\,F_2(x)) \\ & & \quad \wedge V_2(\lambda F_1 \lambda F_2.\,F_2(x)))) \end{array} \right\}$$

$$\texttt{NAM} = \left\{ \begin{array}{rcl} {}_{(1)}N & \mapsto & \lambda E.\,E \\ {}_{(*)}NP & \mapsto & \lambda E \lambda F_1 \lambda F_2.\,\exists x(x = E \wedge F_1(E) \wedge F_2(E)) \end{array} \right\}$$

$$\texttt{EVE} = \left\{ \begin{array}{rcl} {}_{(1)}(S\backslash NP)/S[em] & \mapsto & \lambda E \lambda S \lambda Q.\,Q(\lambda w.\top, \lambda x.\,E(x,S)) \\ {}_{(1)}(S\backslash NP)/S[qem] & \mapsto & \lambda E \lambda S \lambda Q.\,Q(\lambda w.\top, \lambda x.\,E(x,S)) \end{array} \right\}$$

$$\texttt{EXG} = \left\{ \begin{array}{rcl} {}_{(1)}S\backslash NP & \mapsto & \lambda E \lambda Q.\, Q(\lambda w.\, \top, \lambda x.\, \mathbf{prog}(E(x))) \\ {}_{(1)}(S\backslash NP)/NP & \mapsto & \lambda E \lambda Q_1 \lambda Q_2.\, Q_2(\lambda w.\, \top, \lambda x.\, Q_1(\lambda w.\, \top, \lambda y.\, \mathbf{prog}(E(x, y)))) \end{array} \right\}$$

$$\texttt{TNS} = \left\{ \begin{array}{rcl} {}_{(1)}(S[dcl]NP)/PP & \mapsto & \lambda E \lambda F \lambda Q.\, Q(\lambda w.\, \top, F) \\ {}_{(1)}(S\backslash NP)/(S[adj]\backslash NP) & \mapsto & \lambda E \lambda X.\, X \\ {}_{(1)}(S[dcl]\backslash NP)/(S[b]\backslash NP) & \mapsto & \lambda E \lambda V.\, V \end{array} \right\}$$

$$\texttt{PST} = \left\{ \begin{array}{rcl} {}_{(1)}N/N & \mapsto & \lambda E \lambda F \lambda x.\, E(F(x)) \end{array} \right\}$$