

Word Sequence Modeling using Deep Learning: an End-to-end Approach and its Applications

THÈSE N° 7204 (2016)

PRÉSENTÉE LE 28 OCTOBRE 2016

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE L'IDIAP

PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Joël Yvon Roland LEGRAND

acceptée sur proposition du jury:

Dr D. Gillet, président du jury
Prof. H. Bourlard, Dr R. Collobert, directeurs de thèse
Dr C. Gardent, rapporteuse
Dr K. Cho, rapporteur
Dr J.-M. Vesin, rapporteur



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

To my grandfather.

Acknowledgements

I would like to thank my advisor, Ronan Collobert for allowing me to do that thesis and Hervé Bourlard for giving me the opportunity to work at Idiap. I also thank NEC and Facebook, who partially funded this thesis.

I am grateful to my thesis committee, Denis Gillet, Claire Gardent, Kyunghyun Cho and Jean-Marc Vesin for reviewing this thesis and providing constructive comments and suggestions.

A special thanks to my colleagues of the AML group, Dimitri, Remi and Pedro, my brothers in arms. I will always remember the nights we spent at Idiap before the deadlines, our trip to Vegas (baby) and all the amazing time we spend together. Special thanks to Marc who helped me writing and organizing the thesis and to Dimitri who patiently revised and corrected my thesis twice and morally supported me during the many stressful periods.

Finally, thanks to my parents to whom I owe everything and to my wife for his love and support during all these years.

Lausanne, August 29th.

J. L.

Abstract

For a long time, natural language processing (NLP) has relied on generative models with task specific and manually engineered features. Recently, there has been a resurgence of interest for neural networks in the machine learning community, obtaining state-of-the-art results in various fields such as computer vision, speech processing and natural language processing. The central idea behind these approaches is to learn features and models simultaneously, in an end-to-end manner, and making as few assumptions as possible. In NLP, *word embeddings*, mapping words in a dictionary on a continuous low-dimensional vector space, have proven to be very efficient for a large variety of tasks while requiring almost no *a-priori* linguistic assumptions.

In this thesis, we investigate continuous representations of segments in a sentence for the purpose of solving NLP tasks that involve complex sentence-level relationships. Our sequence modelling approach is based on neural networks and takes advantage of word embeddings. A first approach models words *in context* in the form of continuous vector representations which are used to solve the task of interest. With the use of a compositional procedure, allowing arbitrarily-sized segments to be compressed onto continuous vectors, the model is able to consider long-range word dependencies as well.

We first validate our approach on the task of *bilingual word alignment*, consisting in finding word correspondences between a sentence in two different languages. Source and target words *in context* are modeled using convolutional neural networks, obtaining representations that are later used to compute alignment scores. An aggregation operation enables unsupervised training for this task. We show that our model outperforms a standard generative model.

The model above is extended to tackle phrase prediction tasks where phrases rather than single words are to be tagged. These tasks have been typically cast as classic word tagging problems using special tagging schemes to identify the segments boundaries. The proposed neural model focuses on learning fixed-size representations of arbitrarily-sized chunks of words that are used to solve the tagging task. A compositional operation is introduced in this work for the purpose of computing these representations. We demonstrate the viability of the proposed representations by evaluating the approach on the *multiwork expression tagging* task.

Acknowledgements

The remainder of this thesis addresses the task of *syntactic constituency parsing* which, as opposed to the above tasks, aims at producing a structured output, in the form of a tree, of an input sentence. Syntactic parsing is cast as multiple phrase prediction problems that are solved recursively in a greedy manner.

An extension using recursive compositional vector representations, allowing for lexical information to be propagated from early stages, is explored as well. This approach is evaluated on a standard corpus obtaining performance comparable to generative models with much shorter computation time. Finally, morphological tags are included as additional features, using a similar composition procedure, to improve the parsing performance for morphologically rich languages. State-of-the-art results were obtained for these task and languages.

Key words: Neural Networks, Deep Learning, Natural Language Processing, Bilingual Word Alignment, Tagging, Syntactic Parsing

Résumé

Le traitement automatique des langues (TAL) a longtemps été fondé sur des modèles génératifs utilisant des données d'entrée conçues pour refléter nos connaissances linguistiques et spécifiques aux tâches traitées. Récemment, les réseaux de neurones artificiels ont connu un net regain d'intérêt de la part de la communauté scientifique; cela a engendré d'excellents résultats dans des domaines aussi divers que la vision par ordinateur, le traitement de la parole ou le traitement automatique des langues. L'originalité de ces approches réside dans le fait d'apprendre les représentations d'entrée en même temps que les paramètres du modèle, et ce en utilisant le moins de connaissances *a priori* possibles. En TAL, cette idée se traduit par l'utilisation de représentations vectorielles continues de petite dimension appelées "*word embeddings*" et elle s'est révélée être très efficace pour une grande variété de tâches, tout en ne nécessitant que très peu d'hypothèses linguistiques.

Nous étudions dans cette thèse des représentations continues de segments de phrases, dans le but de résoudre des tâches impliquant des relations complexes et distantes entre les mots. L'approche proposée utilise des réseaux de neurones artificiels, prenant comme entrées des représentations continues de mots. Nous présentons dans un premier temps une méthode de modélisation de mots en contexte sous forme de représentations vectorielles continues. Les représentations obtenues sont utilisées pour résoudre diverses tâches de TAL. Nous proposons ensuite une procédure compositionnelle permettant de représenter des segments de différentes tailles sous forme de vecteur de taille fixe. Cette nouvelle procédure permet au modèle de considérer des relations longue distance entre les mots d'une phrase.

Notre approche est tout d'abord validée sur une tâche d'alignement de mots dans un corpus bilingue consistant à établir des correspondances entre les mots d'une phrase exprimée dans deux langues différentes. Nous utilisons pour cela un réseau de neurones convolutionnel dans le but d'extraire des représentations continues de mots en contexte. Ces représentations sont ensuite utilisées pour calculer des scores d'alignement. Une opération d'agrégation permet au modèle d'être entraîné de manière non supervisée. Nous observons que notre modèle obtient des résultats supérieurs à ceux d'un modèle génératif populaire.

Ce modèle est ensuite étendu dans le but de résoudre des problèmes dans lesquels il s'agit d'annoter des phrases plutôt que des mots. Ces problèmes sont généralement perçus comme des problèmes de classification de mots en utilisant un système d'annotation permettant

Acknowledgements

d'identifier les extrémités des segments à annoter. L'approche que nous proposons permet d'extraire des représentations vectorielles de tailles fixes à partir de segments de phrases de tailles variables. Pour obtenir ces représentations, nous introduisons une opération de composition. La viabilité de cette procédure est démontrée empiriquement sur une tâche de de détection et d'annotation d'expressions multi-mots.

La suite de la thèse traite une tâche d'analyse syntaxique. Cette tâche, par opposition aux précédentes, vise à produire une sortie structurée, sous forme d'arbre, à partir d'une phrase d'entrée. Cette analyse est abordée comme une suite de problèmes d'annotation de segments de phrase, résolus récursivement de manière gloutonne.

Nous mettons ensuite en place une procédure récursive tirant parti d'une opération de composition, dans le but de propager de l'information syntaxique et sémantique durant la procédure gloutonne. Une évaluation sur une tâche d'analyse syntaxique standard nous montre que notre approche permet d'obtenir des résultats équivalents à ceux d'un modèle génératif populaire, tout en étant plus rapide. Nous proposons enfin une procédure similaire dont le but est de propager de l'information morphologique, dans le cadre de l'analyse syntaxique de langues morphologiquement riches. Nous parvenons grâce à cette procédure aux meilleurs résultats jamais obtenus pour cette tâche pour la majorité des langues prises en compte.

Mots clefs: Réseaux de Neurones, Apprentissage Profond, Traitement Automatique des Langues, Alignement de Mots dans un Corpus Bilingue, l'Étiquetage, Analyse syntaxique

Contents

Acknowledgements	v
Abstract (English/Français)	vii
List of figures	xv
List of tables	xvii
List of acronyms	xvii
Notations	xxi
1 Introduction	1
1.1 Motivations and Objectives	2
1.2 Contributions	3
1.3 Organization of the thesis	5
2 Background	7
2.1 Machine learning	7
2.1.1 Supervised learning	7
2.1.2 Unsupervised learning	8
2.1.3 Semi-supervised learning	9
2.2 Neural networks	9
2.2.1 Feed forward neural networks	9
2.2.2 Softmax layer	11
2.2.3 Recursive neural networks	11
2.2.4 Convolutional neural networks	12
2.3 Deep Learning for NLP	13
2.3.1 Word embeddings	13
2.3.2 Deep Learning for word sequence processing	15
3 Sequence Processing for Bilingual Word Alignment	21
3.1 Introduction	21
3.2 Related work	22
3.3 Aggregation Model	23

Contents

3.3.1	Unsupervised Training	24
3.3.2	Choosing the Aggregation	25
3.3.3	Decoding	26
3.4	Neural Network Architecture	26
3.4.1	Word embeddings	26
3.4.2	Convolutional layers	27
3.4.3	Additional Features	27
3.5	Experiments	28
3.5.1	Datasets	28
3.5.2	Evaluation	28
3.5.3	Proposed system setup	29
3.5.4	Results	29
3.6	Analysis	32
3.7	Conclusion	33
4	Phrase Prediction: a Chunk-based Approach	35
4.1	Introduction	35
4.2	Related work	36
4.2.1	Phrase prediction	36
4.2.2	Continuous phrase representations	36
4.3	Proposed model	37
4.3.1	Word representation	37
4.3.2	Phrase representation	38
4.3.3	Phrase scoring	38
4.3.4	Structure tag inference	38
4.3.5	Training	39
4.4	Experiments	40
4.4.1	Multiword expression	40
4.4.2	Corpus	40
4.4.3	Evaluation	40
4.4.4	Baseline models	41
4.4.5	Setup	41
4.5	Results	41
4.6	Representation analysis	42
4.7	Conclusion	42
5	Sequence Processing for Structural Inference: Syntactic Parsing	45
5.1	Introduction	45
5.2	Related work	46
5.2.1	State-Of-The-Art	47
5.2.2	Greedy Parsing	47
5.2.3	Syntactic parsing using word embeddings	48
5.3	A greedy discriminative parser	48

5.3.1	Smoothed Context Rule Learning	48
5.3.2	Greedy Recurrent Algorithm	49
5.4	Architecture	50
5.4.1	Words Embeddings	51
5.4.2	Sliding Window BIOES Tagger	52
5.4.3	Aggregating BIOES Predictions	52
5.4.4	Training Likelihood	54
5.5	Experiments	56
5.5.1	Corpus	56
5.5.2	Features	57
5.5.3	Setup	57
5.5.4	Results	57
5.5.5	Rule Prediction Analysis	58
5.6	Conclusion	59
6	RNN-Based Phrase Composition for Syntactic Parsing	61
6.1	Related Work	61
6.2	Greedy RNN Parsing	62
6.2.1	Word-Tag Composition	63
6.2.2	Training Procedure	65
6.3	Experiments	65
6.3.1	Detailed Setup	65
6.3.2	Word Embedding Dropout Regularization	66
6.3.3	Performance comparison	66
6.4	Conclusion	68
7	Syntactic Parsing of Morphologically Rich Language	69
7.1	Introduction	69
7.2	Related work	70
7.3	Recurrent greedy parsing	70
7.4	Parsing Morphologically Rich Languages	71
7.4.1	Morphological features	71
7.4.2	Morphological Embeddings	71
7.4.3	Morphological composition	72
7.5	Experiments	73
7.5.1	Corpus	73
7.5.2	Setup	74
7.5.3	Results	74
7.6	Conclusion	76
8	Conclusion	77
8.1	Future research	79

Contents

Bibliography	93
Curriculum Vitae	95

List of Figures

2.1	Illustration of a 2-layer neural network.	10
2.2	Example of recursive neural network applied on a tree structure.	11
2.3	Illustration of a convolutional layer.	13
2.4	Convolutional architecture for SRL [Collobert et al., 2011].	17
2.5	Simple recurrent neural network [Mikolov et al., 2010].	18
2.6	Recursive neural network for syntactic parsing [Socher et al., 2013a].	19
3.1	Example of word alignment.	22
3.2	Illustration of the alignment model.	24
4.1	Constrained graph for structured inference.	39
5.1	Example on syntactic parse tree.	46
5.2	Illustration of the greedy algorithm.	49
5.3	Sliding window tagger.	53
5.4	Constrained graph for tag inference.	54
5.5	Iterative procedure to generate the training data.	55
5.6	Training corpus pre-processing.	56
5.7	Normalized scores from the network classifier.	59
6.1	Greedy parsing algorithm.	63
6.2	Recurrent composition of a sub-tree.	64
6.3	Train and validation F1-score, according to the number of training iterations, with and without the “dropout” procedure.	66
6.4	Validation F1 and number of sentences, according to the sentence length.	66
7.1	Greedy parsing algorithm.	72
7.2	Sliding window tagger including morphological features.	73
7.3	Recursive composition of the morphological feature <i>gender</i>	74

List of Tables

3.1	Alignment error rates for different aggregation operations in each language direction and with <i>grow-diag-final-and</i> symmetrization.	30
3.2	Alignment error rates using different input features in each language direction and with <i>grow-diag-final-and</i> symmetrization.	30
3.3	English-French results on the test set in terms of precision, recall, F-score and AER.	31
3.4	Romanian-English results.	31
3.5	Czech-English results.	32
3.6	Average BLEU score and standard deviation for five runs of MERT.	32
3.7	Analysis of source window representations.	33
3.8	Analysis of source and target representations.	33
4.1	Number of k-sized chunks in the training corpus	40
4.2	Results on the test corpus in terms of F-measure.	42
4.3	Results on the test corpus in terms of F-measure using an ensemble of 5 models.	42
4.4	Closest neighbors for three input phrases in terms of Euclidean distance.	42
5.1	Simple example of a grammar rule and its corresponding BIOES grammar.	51
5.2	Influence of different features.	58
5.3	Results in terms of Precision, Recall, and F1 score.	58
6.1	Performance comparison of different state-of-the-art parsers, in terms of Precision, Recall, and F1 score.	67
6.2	Detailed parser comparison.	67
6.3	Nearest neighbors for several phrases in the WSJ corpus.	68
7.1	Influence of the additional morphological embeddings in terms of F1-score	75
7.2	Results for all languages in terms of F1-score, using gold POS and morphological tags.	75
7.3	Results for all languages in terms of F1-score using predicted POS and morphological tags.	76

List of Acronyms

AER	alignment error rate
ANN	Artificial neural network
BIOES	Begin Intermediate Other End Single
CFG	Context free grammar
CNF	Chomsky normal form
CNN	Convolutional neural network
CRF	Conditional random fields
CYK	Cocke-Younger-Kasami
EM	Expectation maximization
GRU	gated recurrent unit
HMM	Hidden Markov model
ICA	independent component analysis
LSA	Latent semantic analysis
LSE	LogSumExp
LSTM	Long short-term memory
MLP	Multilayer perceptron
MRL	Morphologically rich language
MWE	multiword expression
NER	Named entity recognition
NLP	Natural language processing
PCA	Principal component analysis
PCFG	Probabilistic context free grammar
POS	Part-of-speech
RNN	Recursive neural network
SPMRL	Syntactic Parsing of MRL
SRL	Semantic role labeling
SRNN	Simple RNN
SVD	Singular value decomposition
WSJ	Wall street journal

Notations

$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	matrices
$\mathbf{a}, \mathbf{b}, \mathbf{c}$	vectors
a, b, c	scalars
$[\mathbf{a} \mathbf{b}]$	concatenation of vector \mathbf{a} and \mathbf{b}
\mathcal{W}	Word dictionary
\mathcal{C}	Character dictionary
\mathcal{T}	Tag dictionary

1 Introduction

Human beings interact with each other using what we call “natural language”, *i.e.* sequences of words expressing concepts willing to be shared. While the term “natural” can give an impression of simplicity, natural language is, in fact, highly complex. First of all, in order to communicate, people need to use the same language, *i.e.* the same vocabulary and the same syntax. Yet, even in these conditions, natural language remains ambiguous in many situations and it requires *a-priori* knowledge about the interlocutors and the context. Misinterpretations and misunderstandings are not only possible but frequent.

The idea of interacting with a computer in natural language quickly raised interest of computer scientists. This interaction requires converting a piece of text into a representation intelligible for the machine. However, despite many efforts from the computational linguistic community, no consensus has emerged on the possibility of an unique representation, nor on what form it should take. Instead, researchers have proposed several specific sub-tasks to extract information reflecting our knowledge about natural language. Each of them consists in extracting semantic information, e.g. in “named entity recognition” and “word sense disambiguation” tasks, as well as syntactic information, e.g. in the “syntactic parsing” and “part-of-speech tagging” tasks. The knowledge obtained is then used to perform higher level natural language processing (NLP) for tasks such as machine translation or information retrieval.

Historically, NLP tasks have been mainly tackled with generative models using task specific and manually engineered features or linear models such as linear support vector machines (SVM) trained over very high-dimensional sparse feature vectors. On the other hand, recent works in NLP using neural networks have focused on learning dense input representations, referred to as embeddings, using minimal *a-priori* knowledge. By learning these representations on large unlabeled databases, these models have been shown to yield state-of-the art performance for various NLP tasks.

Given the success of these methods, this thesis aims at exploring how to take advantage of word embeddings to build continuous representations of sentence segments, in order to solve NLP tasks involving complex relations at the sentence level. We investigate a neural

network-based approach modeling sequences of words in the form of continuous vector representations, that are used to perform the task of interest. Our approach is validated on different natural language processing tasks, both supervised and unsupervised, from simple tagging tasks (multiword expression tagging) to more challenging prediction tasks exploiting structure (constituency parsing).

1.1 Motivations and Objectives

Generative models are at the core of a majority of NLP systems. In the context of classification, they model the joint distribution between the observations and labels (classes). This distribution is often inferred by counting the number of times events occur simultaneously. For instance, language modeling, which is a crucial component in machine translation and information retrieval, still relies on generative models. This task consists in assigning a probability that a sentence is a legal string in a language. This probability is estimated based on the relative frequency of word sequences observed in a training corpus. Similarly, most of the syntactic parsing systems rely on context free grammars (CFG) and assign probabilities to each of the rules by counting their occurrences in a training corpus [Magerman, 1995, Collins, 2003, Charniak, 2000], prior to decoding using chart algorithms. Another example is the machine translation task. The popular IBM machine translation alignment model is generative and based on counting of word co-occurrences [Brown et al., 1990].

Despite being conceptually attractive, generative models face several shortcomings. First, since there are a combinatorial number of possible observations, many rare combinations that are never observed on the training data are assigned zero probability. These models thus need to be carefully smoothed in order to deal with unseen events. Second, generative models tend to make independence assumptions to enable efficient estimation and decoding. This makes difficult to incorporate arbitrary or structured features. Finally, they often rely on task specific features (such as bag-of-words for information retrieval), or hand crafted features carrying linguistic knowledge (such as the head words for syntactic parsing).

On the other hand, discriminative models directly learn the conditional probability distribution of the output classes given the input observations. As they do not model the joint probability between the observations and labels but rather focus on solving a particular task, they tend to obtain better accuracy [Klein and Manning, 2002]. In particular, neural network based models allow for arbitrary input features without the need to make independence assumptions. Nevertheless, it is worth noting that discriminative models also face several issues. They are much more subject to overfitting and they are often more complex than their generative counterparts. They also tend to require more data in order to obtain comparable performance.

Recent advances in machine learning have made it possible to train systems in an end-to-end manner. As discussed in Collobert et al. [2011], this enables minimal use of prior knowledge when applied to NLP. In this study, the authors propose a deep neural network architecture

that learns word representations (the features) and infers tags discriminatively in an end-to-end manner. This architecture is applied to various NLP tasks such as part-of-speech tagging, name entity recognition or semantic role labeling, and it achieves state-of-the-art performance in all of them. More recently, several other works have taken advantage of word embeddings in various NLP domains such as machine translation [Cho et al., 2014b] and question answering [Bordes et al., 2014].

These techniques offer many advantages:

1. they are trained in an end-to-end manner, including the features, by back-propagating the error gradient. This allows to learn relevant representations for the task of interest, while using minimal prior knowledge;
2. they are smoothed by design: even if an example has not been seen in the training corpus, its representation is close to similar examples;
3. these representations can be trained on large unlabeled corpora.

In this thesis, we investigate continuous representations of sentence segments for the purpose of solving NLP tasks involving complex relations at the sentence level. This approach uses word embeddings and models words in context in the form of continuous vector representations. This approach is applied on various NLP tasks from simple tagging tasks (multiword expression detection) to more challenging prediction tasks exploiting structure (constituency parsing). Modeling long-range word dependencies is made possible by the introduction of a novel compositional operation that compute continuous phrase representations of arbitrarily sized segment of sentences.

1.2 Contributions

The contributions of this thesis are summarized as follows:

- Introducing an **unsupervised discriminative word alignment** model for machine translation which outperforms a standard generative baseline on English-Romanian, Romanian-English and Czech-English.

Word alignment is the task of finding the correspondences between words in a pair of sentences that are translations of each other. We propose a neural network model that extracts context information from the source and target sentences and then computes simple dot products to estimate alignment links. The model can be easily trained on unlabeled data via a simple aggregation operation. The aggregation combines the scores of all source words for a particular target word. The network is trained using a soft-margin criterion, which promotes source words which are likely to be aligned with a given target word according to the knowledge the model has learned so far. At test time,

the aggregation operation is removed and source words are aligned to target words by choosing the highest scoring candidates. Results on three different pairs of languages show that our model significantly outperforms a standard generative model. This work has been published in Legrand et al. [2016].

- Investigating a novel **chunk-based approach** for **phrase-prediction**. Phrase prediction problems consist in identifying and labeling phrases in a sentence. These problems are often cast as word classification problems by prefixing every possible tag using the standard BIOES scheme (Begin, Intermediate, Other, End, Single), in order to identify the segment boundaries. A coherent path of tags is then recovered using a constrained transition graph. In this thesis, we propose a novel architecture which models arbitrarily sized chunks into fixed-size representations. These representations are used to perform the chunk classification without going through the intermediate word tagging stage. The architecture is based on neural networks and is trained using a sentence-discriminative objective function based on Conditional Random Fields. We evaluate our approach on the task of multiword expression tagging and compare our performance with a BIOES-based model. We show that the proposed approach performs on par with this state-of-the-art baseline. Furthermore, we show that our system outperforms the system currently obtaining the best results on a standard shared task for MWE tagging. This work has been published in Legrand and Collobert [2016b].

- Introducing a **greedy discriminative constituency parser** using words embeddings.

We propose a bottom-up greedy and purely discriminative syntactic parsing approach that relies only on a few simple features. The core of the architecture is a word sequence modeling architecture which performs a phrase-prediction task, as described in the previous chapter. This is done by leveraging continuous word vector representations to model the conditional distributions of context-aware syntactic rules. The learned distribution rules are naturally smoothed, thanks to the continuous nature of the input features and the model. The tree is built in a greedy manner by recursively applying the model over the new inputs, taking into account the previous node predictions. By successively merging the predicted node to build the new input, our approach allows to reduce the input size and thus enables the model to consider a larger context. Despite the greedy nature of our approach, generalization accuracy compares favorably to existing generative or discriminative non-reranking parsers, while being faster. This work has been published in Legrand and Collobert [2014].

- Introducing a new **compositional procedure** based on **recursive neural networks** performing a summarization of sub-trees in the form of continuous vectors.

We introduce a compositional procedure which outputs a vector representation summarizing sub-trees, both syntactically (parsing/POS tags) and semantically (words). This procedure is jointly trained in an end-to-end manner along with the greedy parser introduced in the previous chapter. The composition is achieved over continuous (word or tag) representations using recursive neural networks. This compositional representation

allows to obtain performance on par with well-known existing parsers, while being faster due to the greedy nature of the parser. We provide a fully functional implementation of this parser¹. This work has been published in Legrand and Collobert [2015].

- Extending the RNN-based greedy parser to **morphologically rich languages** by composing morphological information.

Morphologically rich languages (MRL) are languages in which much of the structural information is contained at the word level, resulting in many forms of word variation. Unlike English, they can have complex word structure as well as flexible word order. We extend our model for syntactic parsing of MRL by learning morphological embeddings. We take advantage of a recursive composition procedure similar to the one used in the previous chapter, to propagate morphological information during the parsing process. We evaluate our approach on the SPMRL (Syntactic Parsing of MRL) Shared Task 2014. We show that integrating morphological features improves performance dramatically, beyond the state-of-the-art for a majority of languages. This work has been published in Legrand and Collobert [2016a].

1.3 Organization of the thesis

The remainder of the thesis is organized as follows:

- Chapter 2, *Background*, presents the machine learning background underlying the whole thesis and introduces the notation used in this work. We then introduce the neural network formalism, including feed-forward, recursive and convolutional neural networks. Finally, we provide a review of the natural language processing literature using deep neural network techniques.
- Chapter 3, *Sequence Processing for Bilingual Word Alignment*, presents the sequence modeling approach, based on convolutional neural networks, applied to the task of bilingual word alignment.
- Chapter 4, *Phrase Prediction: a Chunk-based Approach*, introduces the compositional operation which computes fixed size continuous vectors of arbitrarily sized chunks as well as an extension of the sequence processing approach for phrase prediction applied to the task of multiword expression tagging.
- Chapter 5, *Sequence Processing for Structural Inference: Syntactic Parsing*, introduces the greedy parser which recursively applies a word sequence modeling approach.
- Chapter 6, *RNN-Based Phrase Composition for Syntactic Parsing*, extends the greedy parser introduced in Chapter 5 with the compositional operation introduced in Chapter

¹The parser can be downloaded at joel-legrand.fr/parser.

Chapter 1. Introduction

4 recursively applied to summarize the contents of sub-trees in the form of a continuous vectors.

- Chapter 7, *Syntactic Parsing of Morphologically Rich Language*, extends the greedy parser from Chapter 6 with the morphological composition procedure in the context of morphologically rich languages.
- Chapter 8, *Conclusion*, summarizes the main contributions of this thesis and discusses future research directions.

2 Background

This chapter presents the machine learning background underlying the whole thesis and introduces the notation used in this work. We introduce the neural network formalism, including feed-forward, convolutional and recursive neural networks as well as a review of the natural language processing literature using deep neural networks. Note that the related work specific to each task tackled in this thesis is discussed in the corresponding chapters.

2.1 Machine learning

Machine learning is a field in computer science that explores how machines can learn to solve problems from experimental data rather than being explicitly programmed. The behavior of most machine learning algorithms is conditioned by a set of parameters that define a *model*. The goal of machine learning is to estimate the parameters of this model to learn regular patterns from data observations while avoiding learning the training samples “by heart”. In practice, given a database of training samples an algorithm is expected to learn how to solve a specific task. Note that non-parametric approaches do memorize training examples by heart while generalizing well to unseen examples. This thesis only covers parametric machine learning approaches.

The following section introduces the mathematical formalism that concerns the machine learning algorithms used in the rest of the thesis.

2.1.1 Supervised learning

Supervised learning is a machine learning paradigm that aims at inferring a function from labeled training data. Let \mathcal{X} and \mathcal{Y} be the input and output spaces respectively. We define the set of N training samples $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$. Given a set of functions, $\mathcal{F}: \mathcal{X} \rightarrow \mathcal{Y}$, we define a loss function (the “measure of performance”):

$$Q: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \tag{2.1}$$

Chapter 2. Background

Using this notation, supervised learning consists in finding a function $f \in \mathcal{F}$ which minimizes the empirical risk

$$R = \frac{1}{N} \sum_{n=1}^N Q(f(\mathbf{x}_n), \mathbf{y}_n) \quad (2.2)$$

Supervised learning mainly addresses two types of problems:

Classification, with the output space being defined by a finite set of categories \mathcal{Y} called *classes*. Training consisting in finding a function f which assigns an input vector \mathbf{x} to its corresponding category \mathbf{y} . An example of loss function is:

$$Q(f(\mathbf{x}), y) = \begin{cases} 0 & \text{if } f(\mathbf{x}) = y \\ 1 & \text{otherwise} \end{cases} \quad (2.3)$$

Regression, with the output space being defined as a real vector space \mathcal{Y} . Training consists in finding a function f which minimizes the distance $D(f(\mathbf{x}), \mathbf{y})$, D being an arbitrary metric function. The most common loss function is the Mean Square Error (MSE), defined as

$$Q(f(\mathbf{x}), \mathbf{y}) = \|f(\mathbf{x}) - \mathbf{y}\|^2, \quad (2.4)$$

where $\|\cdot\|$ denotes the L_2 norm.

2.1.2 Unsupervised learning

Unsupervised learning is a machine learning paradigm that aims at discovering hidden structure from unlabeled data. Formally, given a set of N training samples, $\{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$, unsupervised learning finds a function f which minimizes the empirical risk

$$R = \frac{1}{N} \sum_{n=1}^N Q(f(\mathbf{x}_n)) \quad (2.5)$$

where $Q(f(\mathbf{x}))$ is a loss function. Note that unlike supervised learning, Q does not depend on any target label y but only on the input \mathbf{x} .

Unsupervised learning techniques are used to discover intrinsic regularities in data for which no labels are available. They include diverse techniques such as clustering (K-means) and dimensionality reduction (principal component analysis (PCA)). They can be trained in different ways such as predicting the next input in a sequence or minimizing the reconstruction error.

2.1.3 Semi-supervised learning

Semi-supervised learning makes use of unlabeled data along with labeled data. Typically, semi-supervised systems take advantage of large amounts of unsupervised data, which are available at low cost, to overcome the lack of labeled data, requiring expensive human annotations.

2.2 Neural networks

In this thesis, we use artificial neural networks (ANN) as the set of functions \mathcal{F} . Neural networks, as the name indicates, are computational mechanisms inspired by the neural cells in the brain. The following section introduces the notation used in the rest of the thesis for three types of ANN, namely the multi-layer perceptron (MLP), the convolutional neural network (CNN) and the recursive neural network (RNN).

2.2.1 Feed forward neural networks

perceptron is the most elementary neural processing unit. First introduced in Rosenblatt [1957], it mimics a computation unit in a brain. Considering a set of training samples $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_N, y_N)\}$ where the label $y_i \in \{-1, 1\}$, the perceptron is a linear classifier defined as

$$f_{\theta}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \quad (2.6)$$

where $\theta = (\mathbf{w}, b)$ are the parameters to be trained, \mathbf{w} being a real-valued vector of weights and b being a real-valued scalar bias. The training procedure considers every training sample (\mathbf{x}_i, y_i) successively and updates the weights for misclassified samples for which $y_i \cdot f_{\theta}(\mathbf{x}_i) \leq 0$ using the following rules:

$$\mathbf{w} \leftarrow \mathbf{w} + y_i \cdot \mathbf{x}_i \quad b \leftarrow b + y_i \quad (2.7)$$

This update increases the score $y_i \cdot f_{\theta}(\mathbf{x}_i)$. In the case of linearly separable classes, the convergence towards the optimal solution has been proven in Novikoff [1963].

Multi-Layer Perceptrons (MLP) are combinations of perceptron units, organized in successive layers, with non-linear transfer functions being applied at the output of each perceptron. MLPs are known to be universal function approximators [Cybenko, 1989, Hornik et al., 1989]. This means that given a finite number of training samples (\mathbf{x}_i, y_i) and a target function $g(\mathbf{x})$, there exists an MLP that can approximate $g(\mathbf{x})$ with arbitrary precision for all (\mathbf{x}_i, y_i) . Figure 2.1 illustrates this neural network.

While MLPs were historically inspired from neuroscience studies [McCulloch and Pitts, 1988], they can be mathematically modeled using algebraic operations. An MLP can be seen as a

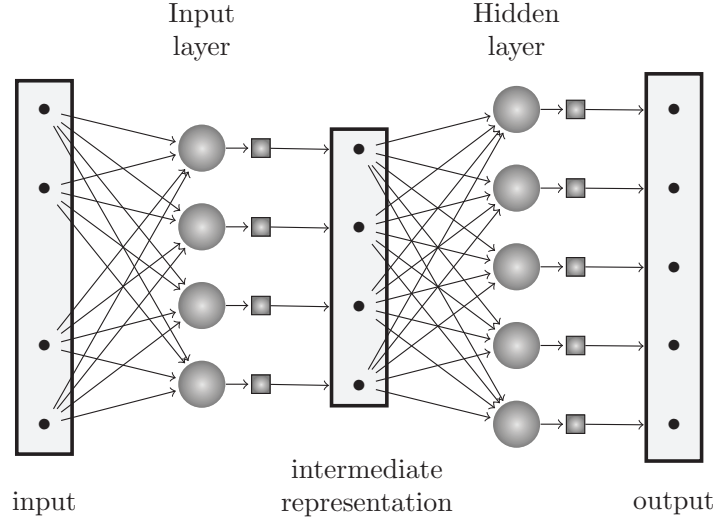


Figure 2.1: Illustration of a 2-layer neural network.

stack of matrix-vector multiplications, followed by point-wise non-linear operations. Formally, an MLP with L layers can be defined as the function

$$f_{\theta}(\mathbf{x}) = \Phi_L(\Phi_{L-1}(\dots \Phi_2(\Phi_1(\mathbf{x})))) \quad (2.8)$$

with

$$\Phi_l(\mathbf{x}) : \mathbb{R}^{d_l^{in}} \rightarrow \mathbb{R}^{d_l^{out}} = h(\mathbf{W}_l \cdot \mathbf{x} + \mathbf{b}_l) \quad (2.9)$$

for layer l , where $\mathbf{x} \in \mathbb{R}^{d_l^{in}}$ is an input vector, $\mathbf{W}_l \in \mathbb{R}^{d_l^{out} \times d_l^{in}}$ a matrix of weights, $\mathbf{b}_l \in \mathbb{R}^{d_l^{out}}$ a vector of biases and $h(\cdot)$ a point-wise non-linear function such as the hyperbolic tangent or the sigmoid function. d_l^{in} and d_l^{out} denote the input and output dimensions of layer l respectively.

Training an MLP: Many techniques exist to train an MLP neural network [Battiti, 1992]. Amongst them, the most popular remains the back-propagation algorithm. First applied to MLP in Le Cun [1985] and Rumelhart et al. [1986], it consists in back-propagating the error at the network output, computed using the loss function Q , towards the previous layers using chained derivatives. In this thesis, we use the stochastic gradient descent [Bottou, 1991] which has proven to be very effective in the case of large-scale machine learning problems [Bousquet and Bottou, 2008]. Gradient descent updates every parameter of the network according to its influence on the criterion error. For this purpose, the partial derivatives Δ_i of the criterion function with respect to every parameter θ_i are computed, as

$$\Delta_i(\mathbf{x}) = \frac{\delta Q(f_{\theta}(\mathbf{x}), y)}{\delta \theta_i} \quad (2.10)$$

Each parameter is updated according to this error derivative:

$$\theta_i \leftarrow \theta_i - \lambda \Delta_i(\mathbf{x}) \tag{2.11}$$

where λ is an update coefficient.

2.2.2 Softmax layer

In the context of classification, it is often useful to obtain the posterior probability of an input vector being assigned a given class. If we denote $f_{\theta}^i(x)$ the i^{th} output of the network, we can give this score a probabilistic interpretation by applying a softmax operation over all possible classes [Bridle, 1990]:

$$P(i|\mathbf{x}, \theta) = \frac{e^{f_i(\mathbf{x})}}{\sum_j e^{f_j(\mathbf{x})}} \tag{2.12}$$

2.2.3 Recursive neural networks

First introduced in Pollack [1990], recursive neural networks (RNN) are neural networks in which the same set of weights is applied recursively over a graph describing a particular structure. They can be applied on arbitrary structures, by using fixed size representations of variable-length input elements. Training is done using the back-propagation through structure (BPTS) algorithm [Goller and Küchler, 1996]. Note that recurrent neural networks are recursive neural networks with the recursion being performed over time.

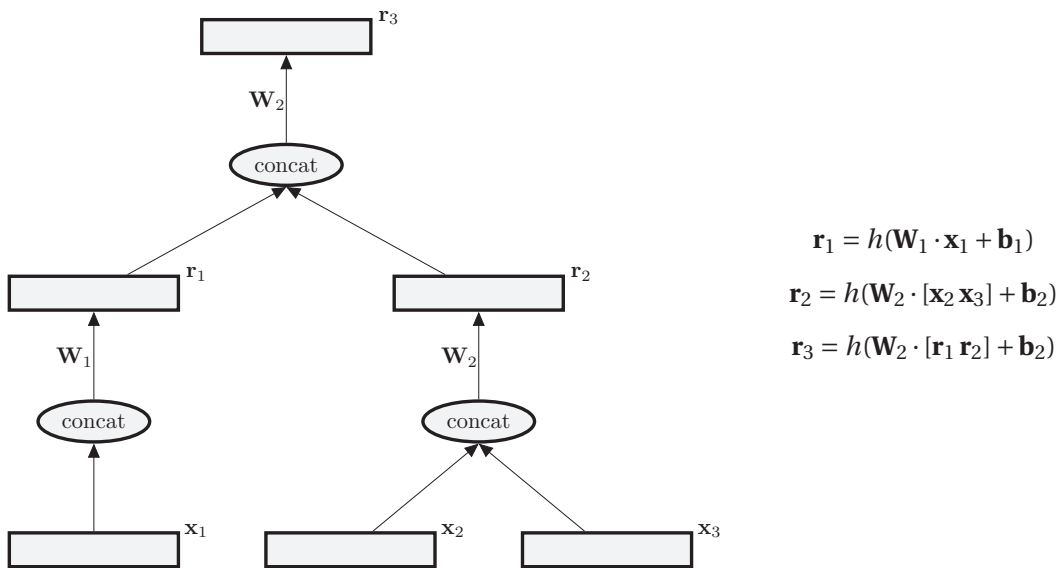


Figure 2.2: Example of recursive neural network applied on a tree structure.

Chapter 2. Background

Figure 2.2 presents an example of structure (a tree) on which a RNN is applied. We define $[\mathbf{x}_1 \mathbf{x}_2]$ the concatenation of vectors \mathbf{x}_1 and \mathbf{x}_2 . \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 are three input vector $\in \mathbb{R}^d$ (where d is the input dimension). \mathbf{r}_1 , \mathbf{r}_2 and \mathbf{r}_3 are three vectorial representations of tree nodes. \mathbf{W}_1 is a matrix of weight of size $d \times d$. \mathbf{W}_2 is a matrix of weight of size $2 \cdot d \times d$. The output of the network is obtained by computing all the node representation recursively, using their child's representations as input. This implies that the node computation order is imposed by the structure of the graph.

2.2.4 Convolutional neural networks

While “classical” linear layers in standard MLPs accept a fixed-size input vector, a convolution layer is assumed to be fed with a sequence of N vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N\}$. A temporal convolutional layer applies the same linear transformation over each successive (or interspaced by d frames) windows of k frames, as illustrated in Figure 2.3. The transformation at frame i is formally written as

$$\mathbf{W} \begin{pmatrix} \mathbf{x}_{i-(k-1)/2} \\ \vdots \\ \mathbf{x}_{i+(k-1)/2} \end{pmatrix}, \quad (2.13)$$

where \mathbf{W} is a $d^{out} \times d^{in}$ matrix of parameters, d^{in} denotes the input dimension and d^{out} denotes the dimension of the output frame.

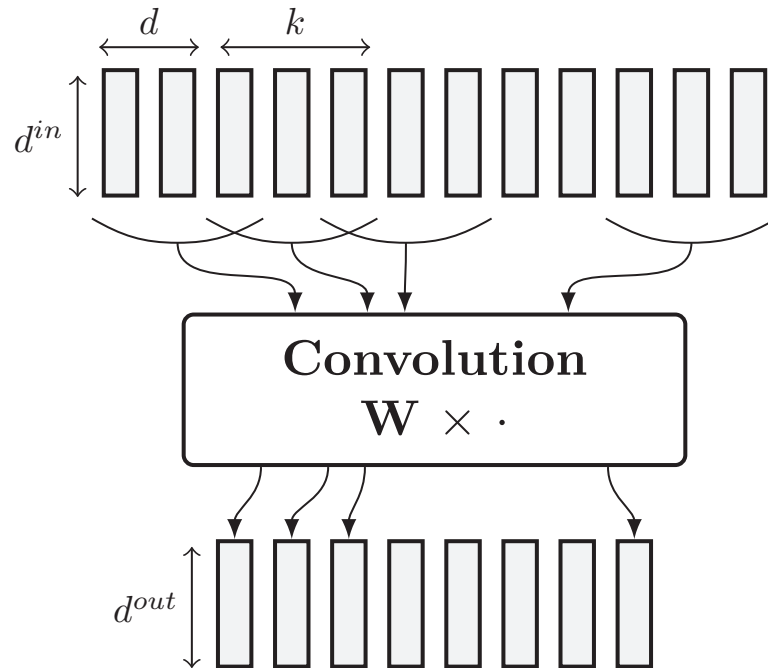


Figure 2.3: Illustration of a convolutional layer.

2.3 Deep Learning for NLP

Deep learning is a branch of machine learning which aims at learning successive levels of representations, using multiple levels of non-linear information processing, from low-level to abstract high-level ones, in order to model complex relationships. These representations are often trained on unlabeled data using unsupervised training techniques to be later used to solve a particular task in a supervised fashion. In NLP, deep learning methods are typically based on neural networks and rely on continuous word vector representations referred to as word embeddings. This section first introduces the concept of word embeddings as well as different unsupervised techniques to produce such representations. NLP applications taking advantage of words embeddings are described at the end of the section.

2.3.1 Word embeddings

Natural language must deal with a large number of words that span a high dimensional and sparse space of possibilities. However, as discussed in Harris [1954], Firth [1957] and Wittgenstein [1953], words that occur in similar contexts tend to have similar meanings. This suggests that the underlying structure of such high dimensional space can be represented in a more compact way. One of the first approaches to capture linguistic knowledge in a low-dimensional space is the Brown clustering algorithm [Brown et al., 1992], grouping words into clusters assumed to be semantically related. A word is then represented by a low-dimensional binary vector representing a path in a binary tree.

Unsupervised training: Word embeddings are usually trained on large unlabeled datasets. Unsupervised training methods generally fall into one of the following categories:

- **Spectral methods:** Several methods based on the spectral decomposition of the word co-occurrence matrix has been developed to produce words embeddings e.g. using the singular value decomposition (SVD) as in latent semantic analysis (LSA) [Landauer and Dumais, 1997] or independent component analysis (ICA) [Väyrynen and Honkela, 2004]. A recent approach [Lebret and Collobert, 2014] proposed a principal component analysis (PCA) of the word co-occurrence probability matrix while minimizing the reconstruction error using the Hellinger distance instead of the usual Euclidean distance. In this last study, low-dimensional representations were evaluated on several NLP tasks showing significant improvements over existing word embeddings.
- **Neural network methods:** A method to learn dense word representations using a neural network based language model was proposed in Bengio et al. [2003]. This work inspired other techniques such as a multi-task learning variant [Collobert and Weston, 2008], showing generalization improvements for several tasks, a hierarchical distributed language model [Mnih and Hinton, 2009] producing better word representations while reducing training time, and the skip-gram model [Mikolov et al., 2013a], a conceptually simple and performing neural network architecture for computing continuous vector representations of words. For the latter, the words surrounding a given input word are predicted from a sentence. All of these methods use large unlabeled corpora as training data.

Many tools either using factorization of word co-occurrence statistics, such as GloVe [Pennington et al., 2014] and HPCA [Lebret and Collobert, 2014] or using neural network models, such as those implemented in word2vec [Mikolov et al., 2013a] have been proposed. Several studies have been conducted in order to distinguish the characteristics of the embeddings obtained using these different publicly released methods. For instance, Chen et al. [2013] proposed several tasks designed to evaluate how well embeddings capture different types of information. It was concluded that, depending on their design, training corpus size and choice of objective function, some embedding techniques perform better than others on certain tasks. They suggested that the application domain should determine the embedding method. Later on, Hill et al. [2014] re-emphasized that no embedding approach is the best for all tasks. Recently, several works have demonstrated both theoretically and empirically the correspondence between spectral and neural network-based methods [Levy and Goldberg, 2014, Pennington et al., 2014].

Evaluation of word embeddings: The first attempts to evaluate the ability of word embeddings to capture linguistic information were mainly qualitative and consisted in manually inspecting the closest embeddings, according to a given metric (e.g. cosine distance, Euclidean distance). Aside from providing a subjective evaluation of the quality of the embeddings, such

method provides a valuable insight into the nature of the learned embeddings. For instance, in Levy and Goldberg [2014], the authors proposed a generalization of the skip-gram model considering syntactic contexts derived from automatically produced dependency parse-trees. Their qualitative study showed that different kinds of contexts produce noticeably different embeddings, and induce different word similarities. In particular, they stated that the bag-of-words nature of the contexts in the “original” skip-gram model yield broad topical similarities (as in lion:zoo), while the dependency-based contexts yield more functional similarities (as in lion:cat).

Quantitative evaluations of word embeddings can be classified into intrinsic and extrinsic tasks. Intrinsic tasks mostly include predicting human judgments of semantic relations between words. For instance, the corpus WordSim-353 [Finkelstein et al., 2001] contains two sets of English word pairs along with human-assigned similarity judgements. Extrinsic tasks include various real NLP tasks, such as coreference resolution and sentiment analysis. While intrinsic evaluation is still widely used, mainly due to ease of use, their correlation with results on extrinsic evaluations is not very reliable [Schnabel et al., 2015, Tsvetkov et al., 2015]. Despite the rapidly growing interest of the NLP community, research on word embeddings is still young. As expressed in Levy et al. [2015], this domain is still lacking controlled variables as well as transparent and reproducible experiments in order to fairly compare the different embeddings methods.

2.3.2 Deep Learning for word sequence processing

Modeling natural language sequences using neural networks and continuous vector representations has a long history. Early work on distributed representations includes Hinton et al. [1986] and Elman [1990, 1991]. More recently, Bengio et al. [2001] was able to outperform n-gram language models in terms of perplexity by training a neural network using continuous word vectors as inputs. This idea was then taken up in Collobert and Weston [2008] to learn word embeddings in an unsupervised manner. They showed that jointly learning these embeddings, and taking advantage of the large amount of unlabeled data in a multitask framework improved the generalization on all the considered tasks obtaining state-of-the-art results. Word embeddings obtained by predicting words given their context tend to capture semantic and syntactic regularities. They have been shown to preserve semantic proximity in the embedded space, leading to better generalization for unseen words in supervised tasks. Such word embeddings have been reported to improve performance on many NLP tasks [Collobert et al., 2011, Turian and Melamed, 2006]. The study in Turian et al. [2010] used unsupervised word representations as extra word features to further improve system accuracy.

This section introduces different approaches for sequence modeling using continuous representations, including convolutional neural networks, recurrent neural networks and recursive neural networks.

Convolutional neural networks for sequence modeling

The order of the words of a sentence are essential for its comprehension. For NLP tasks such as sentiment analysis which consists in identifying and extracting subjective information from pieces of text, taking the word order into account is critical. Classical NLP features such as bag-of-words do not conserve this information and would assign the sentences “it was not good, it was actually quite bad” and “it was not bad, it was actually quite good” the exact same representation.

Convolutional neural networks (CNN), first introduced in the computer vision literature [Lecun, 1989] and described in Section 2.2.4 allow for the extraction of contextual information and focus on relevant information regardless of its position in the input sequence. In NLP, CNN were first introduced by the pioneering work of Collobert et al. [2011] for the task of semantic role labeling. In this task, the tag of a word depends on a verb (or, more correctly, predicate) chosen beforehand in the sentence. The tagging of a word requires the consideration of the whole sentence. As illustrated in Figure 2.4, the authors introduced an architecture that extracts local feature vectors using a convolutional layer. These features are then combined using a pooling operation in order to obtain a global feature vector. The pooling operation is a simple max-over-time operation which forces the network to capture the most useful local features produced by the convolutional layer. This procedure results in a fixed-size representation independent of the sentence length, so that subsequent linear layers can be applied.

Convolutional and pooling architectures have shown promising results on many tasks, including document classification [Johnson and Zhang, 2014], short-text categorization [Wang et al., 2015], sentiment classification [Kalchbrenner et al., 2014], classification of relation type between entities [Zeng et al., 2014, dos Santos et al., 2015], event detection [Chen et al., 2015, Nguyen and Grishman, 2015], paraphrase identification [Yin and Schütze, 2015], question answering [Dong et al., 2015], predicting box revenues of movies based on critic reviews [Bitvai and Cohn, 2015] modeling text interestingness [Gao et al., 2014], and modeling the relation between character-sequences and part-of-speech tags [dos Santos and Zadrozny, 2014].

Recurrent neural networks for sequence modeling

Convolutional networks encode a sequence into a fixed-size vector. However, order sensitivity is constrained to mostly local patterns while disregarding the order of these patterns. On the other hand, recurrent neural networks allow to represent arbitrarily sized linearly structured inputs into a fixed-size vector, while taking the structured properties of the input into account. We can identify three different recurrent neural networks, simple recurrent neural networks (SRNN), long short term memory (LSTM) and gated recurrent units (GRU).

SRNN were first introduced in Elman [1990] and have proven to be effective for sequence modeling tasks. As illustrated in Figure 2.5 a SRNN is a three-layer feed-forward neural

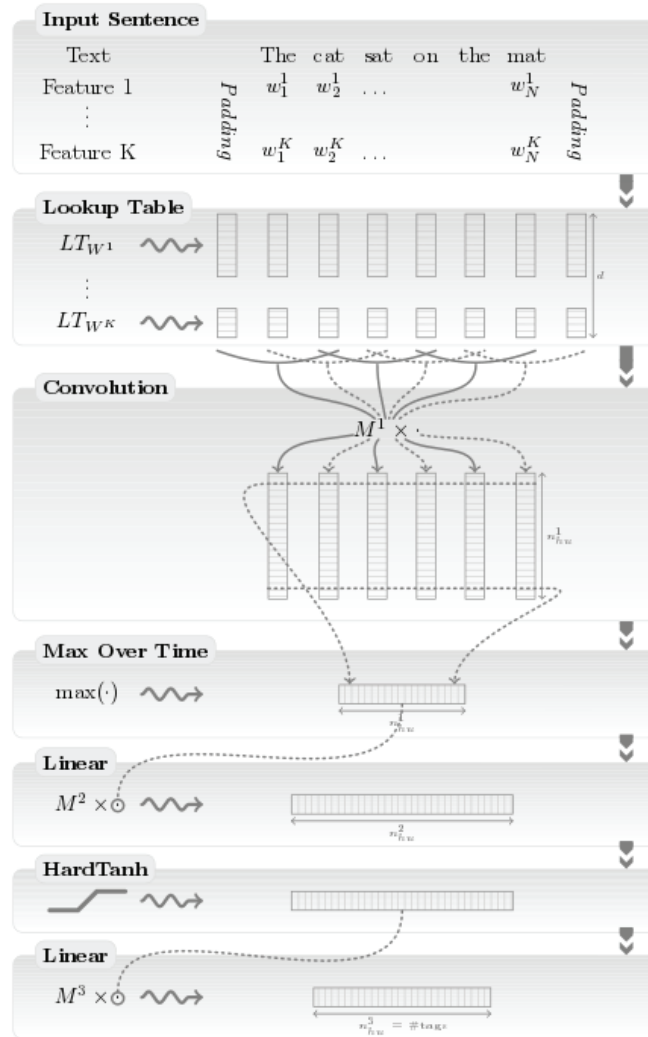


Figure 2.4: Convolutional architecture for SRL [Collobert et al., 2011].

network where a part of the input consists of a copy of the network’s own hidden units at the previous time step. On the task of language modeling, SRNN have been shown to provide better perplexities than traditional models [Mikolov et al., 2010, Mikolov, 2012]. SRNN have been successfully applied to sequence tagging tasks in [Xu et al., 2015, Irsay and Cardie, 2014].

The SRNN is hard to train effectively because of the vanishing gradients problem (see Hochreiter et al. [2001]), making it hard to capture long-range dependencies. Error signals (gradients) back-propagated from the later steps tend to vanish quickly and do not reach earlier input signals. The long short-term memory (LSTM) architecture [Hochreiter and Schmidhuber, 1997] was designed to solve this problem. The main idea is to introduce “memory cells” as part of the state representation that can selectively preserve gradients for an arbitrary length of time. The access to the memory cells is controlled by a smooth mathematical function that simulates logical gates. LSTM were successfully applied to various NLP tasks such as language

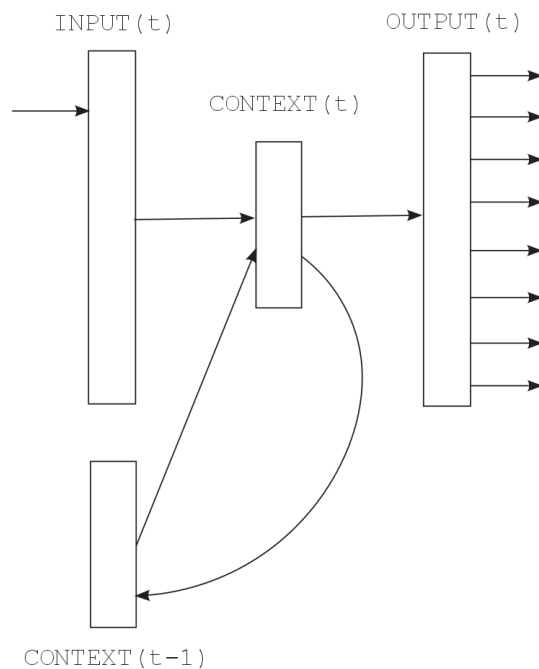


Figure 2.5: Simple recurrent neural network [Mikolov et al., 2010].

modeling [Sundermeyer et al., 2012] and syntactic parsing [Vinyals et al., 2015]. In particular LSTM were shown to be surprisingly effective for machine translation [Sutskever et al., 2014].

While the LSTM architecture is very effective, it is also complex and computationally intensive, making it hard to be analyze [Józefowicz et al., 2015]. The gated recurrent unit (GRU) was recently introduced by Cho et al. [2014b] as an alternative to the LSTM. It was shown to perform comparably to the LSTM on several tasks [Chung et al., 2014]. The GRU was also shown to be effective for machine translation [Cho et al., 2014a].

Recursive neural networks for sequence modeling

While recurrent neural networks are useful for modeling sequences, natural language often requires to take tree structures into account. For example, the syntactic structure of a sentence can be represented as a tree of syntactic relations between sub-constituents. The recursive neural networks (RNN) abstraction introduced in Pollack [1990] is a generalization of recurrent neural networks which allows to deal with arbitrary data structures. In particular, they have been popularized in NLP by the work of Socher et al. [2013a] for syntactic parsing. In this work, the authors learn syntactico-semantic vector representations of tree nodes by recursively applying a compositional operation, following the parse tree. As illustrated in Figure 2.6, the leaves correspond to the sentence words and are assigned a continuous vector representation. Node representations are computed in a bottom-up manner from the leaves to the top tree node. These representations are trained to discriminate the correct parse tree from trees

coming from a generative parser. The system is then used to re-rank the 200-best output of a generative syntactic parser by computing the global score for each tree candidate.

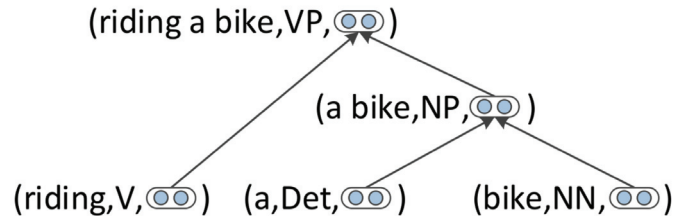


Figure 2.6: Recursive neural network for syntactic parsing [Socher et al., 2013a].

Recursive models were successfully applied to structure prediction tasks such as constituency parse re-ranking [Socher et al., 2013a], dependency parsing [Le and Zuidema, 2014, Chenxi et al., 2015], discourse parsing [Li et al., 2014], semantic relation classification [Hashimoto et al., 2013, Liu et al., 2015] political ideology detection based on parse trees [Iyyer et al., 2014b], sentiment classification [Socher et al., 2013b, Hermann and Blunsom, 2013], target-dependent sentiment classification [Dong et al., 2014] and question answering [Iyyer et al., 2014a].

3 Sequence Processing for Bilingual Word Alignment

In this chapter, we explore the modeling of words in context in the form of continuous vector representations for a bilingual word alignment task. The chapter is organized as follows: We first review the state-of-the-art literature for the task of bilingual word alignment. Further on, we introduce the proposed convolutional neural network-based architecture. We then evaluate several forms of our aggregation operation such as computing the *sum*, *max* and *LogSumExp* over alignment scores. Finally, we provide a comparative evaluation on three standard alignment tasks as well as an analysis of the representations learned by our model.

3.1 Introduction

Bilingual word alignment is the task of finding the correspondences between words in a pair of sentences (the source and the target) that are translations of each other. Word alignment is the first step of a majority of statistical machine translation systems. Even though the best performing systems are phrase-based, the phrase translations are extracted using word alignments most of the time. As illustrated in Figure 3.1, an alignment is a “many to many” correspondence. A source word can be aligned with several target words and vice versa. Furthermore, a source word may not be aligned with any target words. Historically, this task has been mainly tackled using generative models [Brown et al., 1990, Vogel et al., 1996] which still form the basis for many machine translation systems [Koehn et al., 2003, Chiang, 2007]. These models are trained in an unsupervised manner on sentence-aligned corpora although there have been some extensions using small annotated corpora [Och and Ney, 2003].

In this chapter, we introduce a word alignment model based on neural network which extracts context information from the source and target sentences in the form of continuous vector representations. Dot products are computed to estimate alignment links. The model can be easily trained on unlabeled data via a novel but simple *aggregation operation* which has been successfully applied in the computer vision [Pineiro and Collobert, 2015] and speech recognition [Palaz et al., 2016] literatures. The aggregation combines the scores of all source words for a particular target word, and together with the soft-margin criterion, it promotes

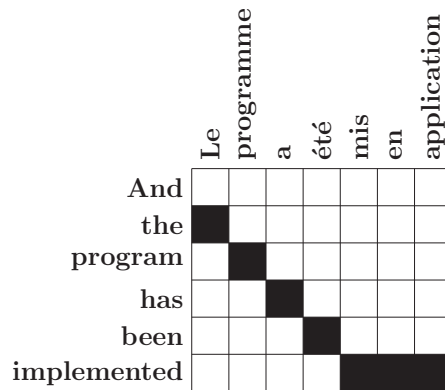


Figure 3.1: Example of word alignment.

source words which are likely to be aligned with a given target word according to the knowledge the model has learned so far. At test time, the aggregation operation is removed and source words are aligned to target words by choosing the highest scoring candidates.

3.2 Related work

Historically, the task of word alignment has been handled by generative models. In Brown et al. [1990], the authors introduced 5 generative models, ranked by increasing order of complexity, each one being built on top of the previous one. IBM model 1 considers a word alignment as a hidden variable with the probability of observed sentence pairs being maximized using the expectation maximization (EM) algorithm. Model 1 does not consider the position of words in a sentence to compute alignments, long-span alignments being as probable as short-span alignments. Model 2 adds a position model to consider different position-dependent alignment probabilities. Model 3 introduces a fertility model that enables one target word to be aligned with several source words. Model 4 and 5 add a relative word order model which allows the alignment to be considered globally, rather than considering each link independently. Vogel et al. [1996] introduced a hidden Markov model (HMM)-based model applied on top of IBM model 3, so that the alignment probabilities are made explicitly dependent on the alignment position of the previous word. This model achieves performance comparable to that of IBM model 4 while being much simpler.

While generative models are still widely use in practice, they face several drawbacks: First, these models can theoretically be trained without supervision. However, Och and Ney [2003] suggested that various parameters, including the probability of jumping to the empty word in the HMM model as well as smoothing parameters for the distortion probabilities and fertility probabilities, should be optimized on annotated data. Second, generative models assume strong independence assumptions between features, making it difficult to incorporate correlated features. To use such features in a generative model, explicit modeling of these dependencies is required. As an example, Toutanova et al. [2002] added part-of-speech tags to

the HMM-based model.

More recently, several discriminative models for word alignment have been proposed. Moore [2005] introduced a discriminative model using a weighted linear combination of a small number of features. The model weights are optimized using a modified version of the averaged perceptron learning as described in Collins [2002]. Taskar et al. [2005] used a large margin approach in which each pair of words is assigned a score reflecting the desirability of the alignment of that pair. The alignment problem is translated into a graph matching problem. Blunsom and Cohn [2006] used a conditional random field (CRF) for this purpose. All of the models above are trained over arbitrary features such as co-occurrence information (Dice coefficient), the distance to the diagonal in the alignment matrix and orthographic features. While they match IBM model 4 performance, these models rely on annotated data. Furthermore, the large-margin and CRF based approaches obtained state-of-the-art results by including IBM model 4 predictions as input features.

In the last few years, several models taking advantage of word embeddings have been proposed. Yang et al. [2013] presented a feed-forward network-based model trained on alignments that were generated by a traditional generative model, potentially considering erroneous alignments as ground truth. Tamura et al. [2014] overcomes this issue by resorting to negative sampling to train a recurrent-neural network on unlabeled data. They both optimize a global loss that requires an expensive beam search decoding procedure to approximate the sum over all alignments. In contrast, our word alignment model is simpler in structure and relies on a more tractable training procedure. Our objective function is word-factored and does not require the expensive computation associated with global loss functions.

3.3 Aggregation Model

In the following, we consider a target-source sentence pair (e, f) , with $e = (e_1, \dots, e_{|e|})$ and $f = (f_1, \dots, f_{|f|})$. Words are represented by f_j and e_i , which are indices in source and target dictionaries. For simplicity, we assume here that word indices are the only feature fed to our architecture. Given a source word f_j and a target word e_i , our architecture embeds a window (of size d_{win}^f and d_{win}^e , respectively) centered around each of these words into a d_{emb} -dimensional vector space. The embedding operation is performed with two distinct neural networks:

$$\text{net}_e([e]_i^{d_{win}^e}) \in \mathbb{R}^{d_{emb}} \quad (3.1)$$

and

$$\text{net}_f([f]_j^{d_{win}^f}) \in \mathbb{R}^{d_{emb}}, \quad (3.2)$$

where we denote the window operator as $[x]_i^d = (x_{i-d/2}, \dots, x_{i+d/2})$. The matching score

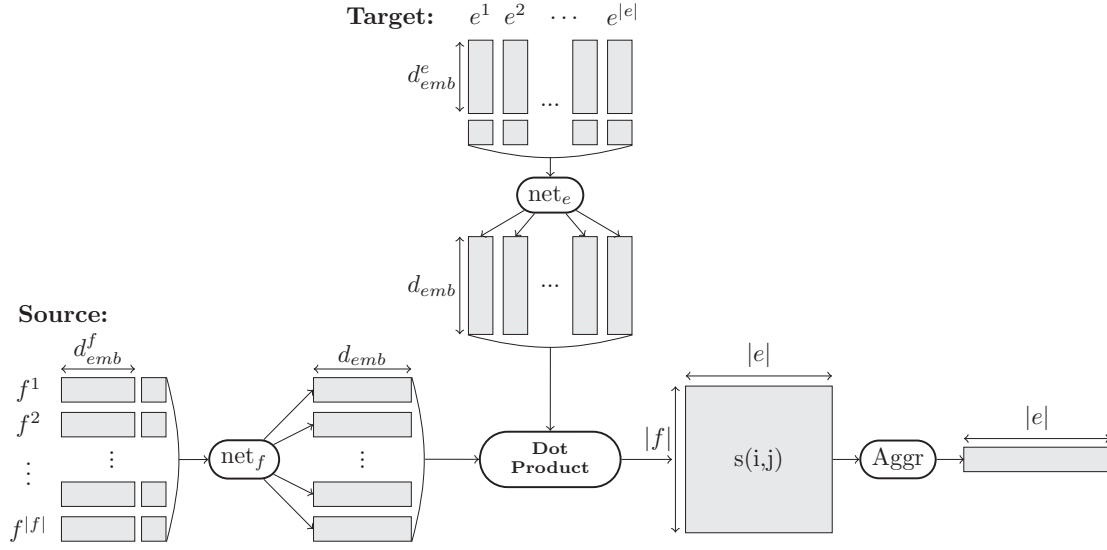


Figure 3.2: Illustration of the alignment model. The two networks net_e and net_f compute representations for source and target words. The score of an alignment link is a simple dot product between those source and target word representations. The aggregation operation summarizes the alignment scores for each target word.

between a source word f_j and a target word e_i is then given by the dot product:

$$s(i, j) = net_e([e]_i^{d_{win}^e}) \cdot net_f([f]_j^{d_{win}^f}). \quad (3.3)$$

If e_i is aligned to f_{a_i} , the score $s(i, a_i)$ should be high, while scores $s(i, j) \forall j \neq a_i$ should be low. Figure 3.2 illustrates the proposed approach.

3.3.1 Unsupervised Training

In this chapter, we consider an unsupervised setup where the alignment is not known at training time. We thus cannot minimize or maximize matching scores (Equation (3.3)) in a direct manner. Instead, given a target word e_i we consider the aggregated matching scores over the source sentence:

$$s_{aggr}(i, f) = \text{Aggr}_{j=1}^{|f|} s(i, j), \quad (3.4)$$

where Aggr is an aggregation operator (see Section 3.3.2). Consider a matching (positive) sentence pair (e^+, f) and a negative sentence pair (e^-, f) . Given a word at index i^+ in the positive target sentence, we want to maximize the aggregated score $s_{aggr}(i^+, f)$ ($1 \leq i^+ \leq |e^+|$) because we know it should be aligned to at least one source word.¹ Conversely, given a word

¹We discuss how we handle unaligned target words in Section 3.3.3. Also, depending on the decoding algorithm the model can be used to predict many-to-many alignments.

at index i^- in the negative target sentence, we want to minimize $s_{aggr}(i^-, f)$ ($1 \leq i^- \leq |e^-|$) because it is unlikely that the source sentence can explain the negative target word. Following these principles, we consider a simple soft-margin loss:

$$\begin{aligned} \mathcal{L}(e^+, e^-, f) = & \sum_{i^+=1}^{|e^+|} \log(1 + e^{-s_{aggr}(i^+, f)}) \\ & + \sum_{i^-=1}^{|e^-|} \log(1 + e^{+s_{aggr}(i^-, f)}). \end{aligned} \quad (3.5)$$

Training is achieved by minimizing Equation (3.5) and by sampling over triplets (e^+, e^-, f) from the training data.

3.3.2 Choosing the Aggregation

The aggregation operation (Equation (3.4)) is only present during training and acts as a filter which aims to explain a given target word e_i by one or more source words. If we had the word alignments, then we would sum over the source words f_j aligned with e_i . However, in our setup alignments are not available at training time, so we must rely on what the model has learned so far to filter the source words. We consider the following strategies:

- **Max:** encourage the best aligned source word f_j , according to what the model has learned so far. In this case, the aggregation is written as:

$$s_{aggr}(i, f) = \max_{j=1}^{|f|} s(i, j). \quad (3.6)$$

- **Sum:** ignore the knowledge learned so far, and assign the same weight to all source words f_j to explain e_i .² In this case, we have

$$s_{aggr}(i, f) = \sum_{j=1}^{|f|} s(i, j). \quad (3.7)$$

- **LSE:** give similar weights to source words with similar scores. This can be achieved with a LogSumExp aggregation operation defined as:

$$s_{aggr}(i, f) = \frac{1}{r} \log \left(\sum_{j=1}^{|f|} e^{r s(i, j)} \right), \quad (3.8)$$

where r is a positive scalar (to be chosen) controlling the smoothness of the aggregation. For small r , the aggregation is equivalent to a *sum*, and for large r , the aggregation acts as a *max*.

²This can be seen by observing that the gradients for all source words are the same.

3.3.3 Decoding

At test time, we align each target word e_i with the source word f_j for which the matching score $s(i, j)$ in (3.3) is highest.³ However, not every target word is aligned, so we consider only alignments with a matching score above a threshold:

$$s(i, j) > \mu^-(e_i) + \alpha \sigma^-(e_i), \quad (3.9)$$

where α is a tunable hyper-parameter, and

$$\mu^-(e_i) = \mathbb{E}_{\{\tilde{e}_k=e_i \in \tilde{e}, \tilde{f}_j^- \in \tilde{f}^-\}} [s(k, j^-)] \quad (3.10)$$

is the expectation over all training sentences \tilde{e} containing the word e_i , and all words \tilde{f}_j^- belonging to a corresponding negative source sentence \tilde{f}^- , and $\sigma^-(e_i)$ is the respective variance.

3.4 Neural Network Architecture

Our model consists of two convolutional neural networks net_e and net_f as shown in Equation (3.3). Both of them take the same form, so we detail only the target architecture.

3.4.1 Word embeddings

The discrete features $[e]_i^{d_{win}^e}$ are embedded into a d_{emb}^e -dimensional vector space via a lookup-table operation as first introduced in Bengio et al. [2001]:

$$\begin{aligned} \mathbf{x}_i^e &= \text{LT}_{\mathbf{W}^e}([e]_i^{d_{win}^e}) \\ &= (\text{LT}_{\mathbf{W}^e}(e_{i-d_{win}^e/2}), \dots, \text{LT}_{\mathbf{W}^e}(e_{i+d_{win}^e/2})), \end{aligned}$$

where the lookup-table operation applied at index k returns the k^{th} column of the parameter matrix \mathbf{W}^e :

$$\text{LT}_{\mathbf{W}^e}(k) = \mathbf{W}_{\bullet, k}^e. \quad (3.11)$$

The matrix \mathbf{W}^e is of size $|\mathcal{W}^e| \times d_{emb}^e$, where \mathcal{W}^e is the target vocabulary, and d_{emb}^e is the word embedding size for the target words.

³This may result in a source word being aligned to multiple target words.

3.4.2 Convolutional layers

The word embeddings output by the lookup-table are concatenated and fed through two successive 1-D convolution layers. The convolutions use a step size of one and extract context features for each word. The kernel sizes k_1^e and k_2^e determine the size of the window $d_{win}^e = k_1^e + k_2^e - 1$ over which features will be extracted by net_e . In order to obtain windows centered around each word, we add $(k_1^e + k_2^e)/2 - 1$ padding words at the beginning and at the end of each sentence.

The first layer cnn^e applies the linear transformation $\mathbf{M}^{e,1}$ exactly k_2^e times to consecutive spans of size k_1^e to the d_{win}^e words in a given window:

$$cnn^e(\mathbf{x}_i^e) = \mathbf{M}^{e,1} \begin{pmatrix} \text{LT}\mathbf{W}^e([e]_{i-a}^{k_1^e}) \\ \vdots \\ \text{LT}\mathbf{W}^e([e]_{i+a}^{k_1^e}) \end{pmatrix}, \quad (3.12)$$

where $a = \lfloor \frac{k_2^e}{2} \rfloor$, $\mathbf{M}^{e,1} \in \mathbb{R}^{d_{hu}^e \times (d_{emb}^e k_1^e)}$ is a matrix of parameters, and d_{hu}^e is the number of hidden units (hu). The outputs of the first layer cnn^e are concatenated to form a matrix of size $k_2^e d_{hu}^e$ which is fed to the second layer:

$$net_e(\mathbf{x}_i^e) = \mathbf{M}^{e,2} \tanh(cnn^e(x_i^e)) \quad (3.13)$$

where $\mathbf{M}^{e,2} \in \mathbb{R}^{d_{emb}^e \times (k_2^e d_{hu}^e)}$ is a matrix of parameters, and the $\tanh(\cdot)$ operation is applied element wise. The parameters \mathbf{W}^e , $\mathbf{M}^{e,1}$ and $\mathbf{M}^{e,2}$ are trained by stochastic gradient descent to minimize the loss (3.5) introduced in Section 3.3.1.

3.4.3 Additional Features

In addition to the raw word indices, we consider two additional discrete features which were handled in the same way as word features by introducing an additional lookup-table for each of them. The output of all lookup-tables was concatenated, and fed to the two-layer neural network architecture.

Distance to the diagonal. This feature can be computed for a target word e_i and a source word f_j :

$$diag(i, j) = \left| \frac{i}{|e|} - \frac{j}{|f|} \right|, \quad (3.14)$$

This feature allows the model to learn that aligned sentence pairs use roughly the same word order and that alignment links remain close to the diagonal. We use this feature only for the source network because it encodes relative position information which only needs to be encoded once. If we would use absolute position instead, then we would need to encode this information both on the source and the target side.

Part-of-speech Words pairs that are good translations of each other are likely to carry the same part of speech in both languages [Melamed, 1995]. We therefore add the part-of-speech information to the model.

Char n-gram. We consider unigram character position features. Let K be the maximum size for a word in a dictionary. We denote the dictionary of characters as \mathcal{C} . Every character is represented by its index c (with $1 < c < |\mathcal{C}|$). We associate every character c at position k with a vector at position $((k - 1) \cdot |\mathcal{C}|) + c$ in a lookup-table. For a given word, we extract all unigram character position embeddings, and average them to obtain a character embedding for a given word.

3.5 Experiments

3.5.1 Datasets

We use the English-French Hansards corpus as distributed by the NAACL 2003 shared task [Mihalcea and Pedersen, 2003]. This dataset contains 1.1M sentence pairs and the test and validation sets contain 447 and 37 examples respectively. We also evaluate on the Romanian-English dataset of the ACL 2005 shared task [Martin et al., 2005] comprising 48K sentence pairs for training, 248 for testing and 17 for validation. For English-Czech experiments, we use the WMT news commentary corpus for training (150K sentence pairs) and a set of 515 sentences for testing [Bojar and Prokopová, 2006].

3.5.2 Evaluation

Our models are evaluated in terms of precision, recall, F-measure and alignment error rate (AER). We train models in each language direction and then symmetrize the resulting alignments using either the *intersection* or the *grow-diag-final-and* heuristic [Och and Ney, 2003, Koehn et al., 2003]. We validated the choice of symmetrization heuristic on each language pair and chose the best one for each model considering the two aforementioned types as well as *grow-diag-final* and *grow-diag*.

Additionally, we train phrase-based machine translation models with our alignments using the popular Moses toolkit [Koehn et al., 2007]. For English-French, we train on the news commentary corpus v10, for English-Czech we used news commentary corpus v11, and for Romanian-English we used the Europarl corpus v8. We tuned our models on the WMT2015 test set for English-Czech as well as for Romanian-English; for English-French we tuned on

the WMT2014 test set. Final results are reported on the WMT2016 test set for English-Czech as well as Romanian-English, and for English-French we report results on the WMT2015 test set (as there is no track for this language-pair in 2016). We compare our model to Fast Align, a popular log-linear reparameterization of IBM Model 2 [Dyer et al., 2013].

3.5.3 Proposed system setup

The kernel sizes of the target network $\text{net}_e(\cdot)$ are set to $k_1^e = k_2^e = 3$ for all language pairs. The kernel sizes of the source network $\text{net}_f(\cdot)$ are set to $k_1^f = k_2^f = 3$ for Romanian-English as well as English-Czech; and for English-French we used $k_1^f = k_2^f = 1$.

The number of hidden units are $d_{hu}^e = d_{hu}^f = 256$ and d_{emb} is set to 256, The source \mathcal{W}_f and target \mathcal{W}_e dictionaries consist of the 30K most common words for English, French and Romanian, and 80K for Czech. All other words are mapped to a unique *UNK* token. The word embedding sizes d_{emb}^e and d_{emb}^f , as well as the char-n-gram embedding size is 128. For LSE, we set $r = 1$ in Equation (3.8).

We initialize the word embeddings with a PCA computed over the matrix of word co-occurrence counts [Lebret and Collobert, 2014]. The co-occurrence counts were computed over the common crawl corpus provided by WMT16. For part of speech tagging we used the Stanford parser on English-French data, and MarMoT [Mueller et al., 2013] for Romanian-English as well as English-Czech.

We trained 4 systems for the ensembles, each using a different random seed to vary the weight initialization as well as the shuffling of the training set. We averaged the alignment scores predicted by each system before decoding. The alignment threshold variables $\mu^-(e_i)$ and $\sigma^-(e_i)$ for decoding (see Section 3.3.3) were estimated on 1000 random training sentences, using 100 negative sentences for each of them. Words not appearing in this training subset were assigned $\mu^-(e_i) = \sigma^-(e_i) = 0$.

For systems where $d_{win}^e > 1$ and $d_{win}^f > 1$, we saw a tendency of aligning frequent words regardless on if they appeared in the center of the context window or not. For instance, a common mistake would be to align "the *cat* sat", with "PADDING *le* chat". To prevent such behavior, we occasionally replaced the center word in a target window by a random word during training. We do this for every second training example on average and we tuned this rate on the validation set.

3.5.4 Results

We first explore different choices for the aggregation operators described in Section 3.3.2, followed by an ablation to investigate the impact of the different additional features described in Section 3.4.3. Next we compare to the Fast Align baseline. Finally, we evaluate our alignments within a full translation system for all language pairs.

Aggregation operation

Table 3.1 shows that the LogSumExp (LSE) aggregator performs best on all datasets for every direction as well as in the symmetrized setting using the grow-diag-final heuristic. All results are based on a single model trained with the 'distance to the diagonal' feature detailed above.⁴ We therefore use LSE for the remaining experiments.

	Max	Sum	LSE
En-Fr	18.1	23.0	15.1
Fr-En	20.7	26.9	15.8
symmetrized	14.8	24.1	12.8
Ro-En	42.2	42.0	37.6
En-Ro	40.4	40.2	35.7
symmetrized	36.4	35.6	32.2
En-Cz	27.9	35.6	24.5
Cz-En	26.5	33.6	24.5
symmetrized	21.8	32.7	21.0

Table 3.1: Alignment error rates for different aggregation operations in each language direction and with *grow-diag-final-and* symmetrization.

Additional features

Table 3.2 shows the effect of the different input features. Both POS and the distance to the diagonal feature significantly improve accuracy. Position information via the 'distance to the diagonal' feature is helpful for all language pairs, and POS information is more effective for Romanian-English and English-Czech which involve morphologically rich languages. We use the POS and 'distance to the diagonal feature' for the remaining experiments.

	English-French			Romanian-English			English-Czech		
	En-Fr	Fr-En	sym	Ro-En	En-Ro	sym	En-Cz	Cz-En	sym
words	22.2	24.2	15.7	47.0	45.5	40.3	36.9	36.3	29.5
+ POS	20.9	23.9	15.3	45.3	42.9	36.9	35.6	33.7	28.2
+ diag	15.1	15.8	12.8	37.6	35.7	32.2	24.8	24.5	21.0
+ POS + diag	13.2	12.1	10.2	33.1	32.2	27.8	24.6	22.9	19.9

Table 3.2: Alignment error rates using different input features in each language direction and with *grow-diag-final-and* symmetrization.

Comparison with the baseline

In the following results we label our model as NNSA (neural network score aggregation). On English-French data (Table 3.3) our model outperforms the baseline [Dyer et al., 2013] in each

⁴We use kernel sizes $k_1^e = k_2^e = 3$ and $k_1^f = k_2^f = 1$ for all language pairs in this experiment.

individual language direction as well as for the symmetrized setting. With an ensemble of four models, we outperform the baseline by 1.7 AER (from 11.4 to 9.7), and with an individual model we outperform it by 1.2 AER (from 11.4 to 10.2). Note that the choice of symmetrization heuristic greatly affects accuracy, both for the baseline and NNSA.

	P	R	F1	AER
English-French				
Baseline	49.6	89.8	63.9	16.7
NNSA	64.7	80.7	71.8	13.2
+ ensemble	61.5	85.8	71.6	11.6
French-English				
Baseline	52.9	88.4	66.2	16.2
NNSA	61.7	86.3	72.0	12.1
+ ensemble	62.6	86.7	72.7	11.6
symmetrized				
Baseline (inter)	69.6	84.0	76.1	11.4
NNSA (gdfa)	60.4	88.5	71.8	10.2
+ ensemble	59.3	89.9	71.4	9.7

Table 3.3: English-French results on the test set in terms of precision (P), recall (R), F-score (F1) and AER; ensemble denotes a combination of four systems and we use the *intersection* (inter) and *grow-diag-final-and* symmetrization (gdfa) heuristics.

On Romanian-English (Table 3.4) our model outperforms the baseline in both directions as well. Adding ensembles further improves accuracy and leads to a significant improvement of 6 AER over the best symmetrized baseline result (from 32 to 26).

	P	R	F1	AER
Romanian-English				
Baseline	70.0	61.0	65.2	34.8
NNSA	75.1	65.2	69.8	30.2
+ ensemble	75.8	62.8	68.7	31.3
English-Romanian				
Baseline	71.3	60.8	65.6	34.4
NNSA	78.1	61.7	69.0	31.1
+ ensemble	78.4	63.2	70.0	30.0
symmetrized				
Baseline (gdfa)	69.5	66.5	68.0	32.0
NNSA (gdfa)	74.1	71.8	73.0	27.0
+ ensemble	73.0	74.5	73.7	26.0

Table 3.4: Romanian-English results (cf. Table 3.3).

On English-Czech (Table 3.5) our model outperforms the baseline in both directions as well. We added the character feature to better deal with the morphologically rich nature of Czech and the feature reduced AER by 2.1 in the symmetrized setting. An ensemble improved

accuracy further and led to a 7 AER improvement over the best symmetrized baseline result (from 22.8 to 15.8).

	P	R	F1	AER
English-Czech				
Baseline	68.4	73.3	70.7	26.6
NNSA	72.0	74.3	73.1	24.6
+ char n-gram	73.8	75.4	74.6	23.2
+ ensemble	78.8	77.2	78.0	20.0
Czech-English				
Baseline	68.6	74.0	71.2	25.7
NNSA	74.1	74.0	74.0	22.9
+ char n-gram	78.1	74.1	76.1	21.4
+ ensemble	79.1	77.7	78.4	18.7
symmetrized				
Baseline (inter)	88.1	66.6	76.0	22.8
NNSA (gdfa)	75.7	80.3	76.3	19.9
+ char n-gram	76.9	81.3	79.1	17.8
+ ensemble	78.9	83.2	81.0	15.8

Table 3.5: Czech-English results (cf. Table 3.3).

BLEU evaluation

Table 3.6 presents the BLEU evaluation of our alignments. For each language-pair, we select the best alignment model reported in Tables 3.3, 3.4 and 3.5, and align the training data. We use the alignments to run the standard phrase-based training pipeline using those alignments. Our BLEU results show the average BLEU score and standard deviation for five runs of minimum error rate training (MERT; Och 2003).

Our alignments achieve slightly better results for Romanian-English as well as English-Czech while performing on par with Fast Align on English-French translation.

	Baseline	NNSA
French-English	25.4 ± 0.1	25.5 ± 0.1
Romanian-English	21.3 ± 0.1	21.6 ± 0.1
Czech-English	17.2 ± 0.1	17.6 ± 0.1

Table 3.6: Average BLEU score and standard deviation for five runs of MERT.

3.6 Analysis

In this section, we analyze the word representations learned by our model. We first focus on the source representations: given a source window, we obtain its distributional representation and then compute the Euclidean distance to all other source windows in the training corpus.

Table 3.7 shows the nearest windows for two source windows; the closest windows tend to have similar meanings.

the voting process	in working together
the voting area	for working together
the voting power	with working together
the voting rules	from working together
the voting system	about working together
the voting patterns	by working together
the voting ballots	and working together
their voting patterns	while working together

Table 3.7: Analysis of source window representations. Each column shows a window over the source sentence followed by several close neighbors in terms of Euclidean distance (among the 30 nearest).

We then analyze the relation between source and target representations: given a source window we compute the alignment scores for all target sentences in the training corpus. Table 3.8 shows for two source windows which target words have the largest alignment scores. The example "in working together" is particularly interesting since the aligned target words *collabore*, *coordonés*, and *concertés* mean *collaborate*, *coordinated*, and *concerted*, which all carry the same meaning as the source window phrase.

the voting process	in working together
vote	travaillé
voteraient	travailleront
votent	collaboration
voter	travaillant
votant	oeuvrant
scrutin	concertés
suffrage	coordonés
procédure	concert
investiture	collabore
élections	coopération

Table 3.8: Analysis of source and target representations. Each column shows a source window and the target words which are most aligned according to our model.

3.7 Conclusion

In this chapter we presented a simple neural network alignment model trained on unlabeled data. The proposed architecture computes alignment scores as dot products between representations of windows around target and source words. We apply an *aggregation operation* borrowed from the computer vision literature to make unsupervised training possible. The aggregation operation acts as a filter over alignment scores and allows us to determine which

Chapter 3. Sequence Processing for Bilingual Word Alignment

source words explain a given target word. We improve over a popular log-linear reparameterization of IBM Model 2 [Dyer et al., 2013] by up to 6 AER on Romanian-English, 7 AER on English-Czech data and 1.7 AER on English-French alignment. Furthermore, we evaluated the proposed approach on a full machine translation task and showed that our alignment led to significant improvements in terms of BLEU score compared to the baseline.

4 Phrase Prediction: a Chunk-based Approach

In the previous chapter we introduced a neural network architecture which extracts context information from sentences, in the form of continuous vector representations. This is done by applying a convolutional network which, given a fixed size input context around a given word, outputs a fixed-size continuous representation. In this chapter, we investigate continuous representations for arbitrarily-sized sentence segments. We introduce a model that takes advantage of the proposed representations to address a phrase tagging task.

The chapter is organized as follows: we first introduce the concept of phrase tagging. We then review the existing approach for phrase tagging as well as existing methods to obtain continuous phrase representations. We finally introduce a new model for phrase tagging as well as a comparative evaluation in the context of multiword expression tagging.

4.1 Introduction

Traditional NLP tasks such as part-of-speech (POS) tagging or semantic role labeling (SRL) consist in tagging each word in a sentence with a tag. Another class of problems such as Named Entity Recognition (NER) or shallow parsing (chunking) consists in identifying and labeling phrases (*i.e.* groups of words) with predefined tags. Such tasks can be expressed as word classification problems by identifying the phrase boundaries instead of directly identifying the whole phrases. In practice, this consists in prefixing every tag with an extra-label indicating the position of the word inside a phrase (at the beginning (B), inside (I), at the end (E), single word (S) or not in a phrase (O)). Different schemes have been used in the literature, such as the IOB2, IOE1 and IOE2 schemes [Sang and Veenstra, 1999] or BIOES scheme [Uchimoto et al., 2000] with no clear predominance.

In this chapter, we propose to learn fixed-size continuous representations of arbitrarily-sized chunks by composing word embeddings. These representations are used to directly classify phrases without using the classical IOB(ES) prefixing step. The proposed approach is evaluated on the task of multiword expression (MWE) tagging. Using the SPRML 2014 data for French

MWE tagging [Seddah et al., 2013], we show that our phrase representations can capture enough knowledge to perform on par with the BIOES-based model of Collobert et al. [2011] applied to MWE tagging. Furthermore, we show that our system outperforms the winner of the SPMRL (syntactic parsing of morphologically rich language) 2013 shared task for MWE tagging [Constant et al., 2013] which is currently the best published system.

4.2 Related work

In the following sections, we review the literature on phrase prediction and phrase representation.

4.2.1 Phrase prediction

Phrase classification problems have been tackled using various machine learning methods such as Support Vector Machines (SVM) for POS tagging [Giménez and Màrquez, 2004] or chunking [Kudoh and Matsumoto, 2000], second order random fields for chunking [Sun et al., 2008] or a combination of different classifiers for NER [Radu et al., 2003]. All these approaches use carefully selected hand-crafted features. More recently, several studies introduced neural network-based systems that can be trained in an end-to-end manner, using minimal prior knowledge. These models take advantage of continuous representations of words. For example, in Collobert et al. [2011] the authors proposed a deep neural network, which learns the word representations (the features) and produces scores for BIOES-prefixed tags. Their system is trained discriminatively in an end-to-end manner, using a conditional random field [Lafferty et al., 2001] which allows the structure of the sentence to be taken into account. This architecture has been applied to various NLP tasks, like POS tagging, NER or semantic role labeling, and achieves state-of-the-art performance in all of them. As opposed to the proposed approach, all the methods above cast the phrase prediction problem as a classic word tagging problem using special tagging schemes to identify the segments boundaries. Kim [2014] introduces a CNN-based model for sentence classification which applies multiple filters (with varying window sizes) in order to obtain multiple features that are used to perform the sentence classification task. This approach is similar to the one proposed in this thesis in that it models arbitrarily-sized segments in a fixed-size vector space.

4.2.2 Continuous phrase representations

Given the success of word embeddings for NLP, several techniques have been proposed to combine them in order to obtain phrase representations. Several models based on vector addition or point-wise multiplication have been explored in [Mitchell and Lapata, 2010, Blacoe and Lapata, 2012]. Such simple compositions have shown to perform competitively on the paraphrase detection and phrase similarity tasks. More sophisticated approaches used techniques from logic, category theory, and quantum information [Clark et al., 2008]. Other

works used the syntactic relations between words to treat certain words as functions and other as arguments such as adjective-noun composition or noun-verb composition [Baroni and Zamparelli, 2010, Grefenstette et al., 2013]. Recursive neural network models have been introduced for syntactic parsing [Socher et al., 2011b, 2013a] and sentiment classification [Socher et al., 2013b]. In these models, word representations are composed using matrix-vector operations, following a syntactic parse tree. Cho et al. [2014b] used recurrent neural networks in the context of machine translation. One recurrent neural network is used to encode a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols. Both are jointly trained in an end-to-end manner. Mikolov et al. [2013b] extended the skip-gram model from Mikolov et al. [2013a] by first identifying a large number of phrases using a data-driven approach, and then treating the phrases as individual tokens during training.

4.3 Proposed model

The proposed model computes fixed-size continuous vectors of arbitrarily sized chunks which are then used as inputs to a classifier. This is done by projecting every possible window of sizes from 1 to K (K being the maximum size) in a common vector space (the same for all k), using a different neural network for each size k . The representations obtained are given to a classifier which output a score for every possible tag. To ensure that a word belongs to one chunk at most, decoding is done using a structured graph decoding, using the Viterbi algorithm.

4.3.1 Word representation

Given an input sentence $S = \{w_1, \dots, w_N\}$, each word is embedded into a D -dimensional vector space by applying the lookup-table operation described in Section 3.4.1:

$$LT_{\mathbf{W}}(w_n) = \mathbf{W}_{w_n} \tag{4.1}$$

where the matrix $\mathbf{W} \in \mathbb{R}^{D \times |\mathcal{W}|}$ represents the parameters of the lookup layer. Each column $\mathbf{W}_n \in \mathbb{R}^D$ corresponds to the vector embedding of the n^{th} word in the dictionary \mathcal{W} .

Adding additional features (such as part-of-speech tags) can be done by adding a different lookup table for each discrete feature. The input becomes the concatenation of the outputs of all these lookup-tables. For simplicity, we consider only one lookup-table in the rest of the architecture description.

4.3.2 Phrase representation

We denote k -window a window of size $k \in [1, K]$ where K is the maximum window size. Phrase representations for all k -windows within a given sentence are produced by looking, for all sizes from 1 to K , at all successive windows of text, sliding over the sentence, from position 1 to $N - K + 1$. Formally, if we denote

$$\begin{aligned} \mathbf{x}_{n,k} = & [LT_{\mathbf{W}}(w_{n-c}), \dots, LT_{\mathbf{W}}(w_n) \\ & , \dots, \\ & , LT_{\mathbf{W}}(w_{n+k-1}), \dots, LT_{\mathbf{W}}(w_{n+k-1+c})] \end{aligned} \quad (4.2)$$

the concatenated word representations corresponding to the n^{th} k -window (c being the context from each side of the the k -window), its representation is given by

$$\mathbf{r}_{n,k} = \mathbf{M}_k^1 \mathbf{x}_{n,k}, \quad (4.3)$$

where $\mathbf{M}_k^1 \in \mathbb{R}^{(k+2c)D \times nh_u}$ is a matrix of parameters and nh_u the dimension of the phrase representations (which is the same for all k). Words outside the sentence boundaries are assign a special "PADDING" embedding.

4.3.3 Phrase scoring

We denote \mathcal{T} the set of tag and \mathcal{T}_k the set of tags for a k -window. We denote $t_k \in \mathcal{T}_k$ the tag $t \in \mathcal{T}$ for a k -window. The scores for all k -windows are computed by a linear layer, using their corresponding representations as input. Formally, the score for the n^{th} k -window are given by

$$s_{n,k} = \tanh(\mathbf{M}^2 r_{n,k}), \quad (4.4)$$

where $\mathbf{M}^2 \in \mathbb{R}^{nh_u \times |\mathcal{T}|}$ is a matrix of parameters. We define s_{n,t_k} the score for the tag $t_k \in \mathcal{T}_k$ starting at the position $n < N - k + 1$.

4.3.4 Structure tag inference

The scoring layer outputs a matrix of $|\mathcal{T}_k| \times (N - k + 1)$ scores for each window size $k \in K$. The next module (see Figure 4.1) of our system is a structured graph G constrained in order to ensure that a word is tagged only once. Each node G_{n,t_k} is assigned the score s_{n,t_k} (the score of

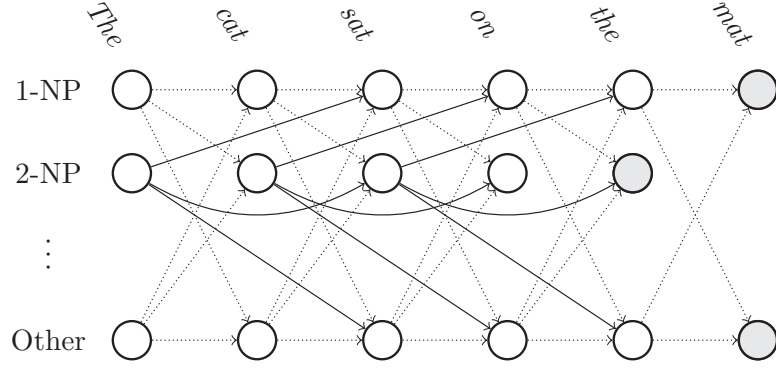


Figure 4.1: Constrained graph for structured inference. Each node is assigned a score from the scoring layer. For instance, the first node of the line 2-NP correspond to the score for the tag NP for the phrase "the cat". Nodes in gray represent final nodes.

the tag $t_k \in \mathcal{T}_k$ starting at the position $n < N - k + 1$) from the scoring layer. Only transitions from a node G_{n,t_k} to a node G_{n+k,t'_k} (with $n + k \leq N$) are possible since a word cannot be tagged twice in the same path. The Viterbi algorithm is then the natural choice to find the best path in the lattice. The score for a sentence S of size N along a path of tags $[t]_1^{N_t}$ is then given by the sum of the tag scores:

$$s(S, [t]_1^{N_t}, \theta) = \sum_{n=1}^{N_t} s_{n,t_k} \quad (4.5)$$

where θ represents all the trainable parameter.

4.3.5 Training

The proposed neural network is trained by maximizing the likelihood over the training data, using stochastic gradient ascent. The score $s(S, [t]_1^{N_t}, \theta)$ can be interpreted as a conditional probability by exponentiating this score and normalizing it with respect to all possible path scores. Taking the log, the conditional probability of the true path $[t]_1^{N_t}$ is given by

$$\log p(s(S, [t]_1^{N_t}, \theta)) = s(S, [t]_1^{N_t}, \theta) - \log\left(\sum_u s(S, [u]_1^{N_u}, \theta)\right) \quad (4.6)$$

Following Rabiner [1990], the normalization term, i.e. the second term of this equation, can be computed in linear time thanks to a recursion similar to the Viterbi algorithm. The whole architecture, including the input feature, phrase representations and scoring layer, is trained through the graph in order to encourage valid paths of tags during training, while discouraging

all other paths.

4.4 Experiments

In this section, we introduce the task of multiword expression detection used to evaluate the proposed approach followed by a comparative evaluation.

4.4.1 Multiword expression

Multiword expressions are groups of tokens which act as single units at some level of linguistic analysis. They cover a wide range of linguistic constructions such as idioms ("kick the bucket"), noun compound ("traffic light") or fixed phrases ("ad hoc"). As they can carry meaning which can not be derived directly from the meaning of individual constituents (as for idioms), they are difficult to handle by automatic systems and represent a key issue for many NLP systems addressing tasks such as machine translation or text generation.

4.4.2 Corpus

Experiments were conducted on the SPMRL french corpus provided for the Shared Task 2013 [Seddah et al., 2013]. This dataset provides 14.7k sentences (443k tokens) with 22.6k identified MWE. A given MWE is defined as a continuous sequence of terminals, plus a POS tag among the 10 possible POS tags. As presented in Table 4.1, a wide majority of the chunks are 2-chunks or 3-chunks (91.2%).

Chunk size	2	3	4	5	5+
#chunk	11108	10188	1702	309	250
percentage	47.2	43.2	7.2	1.3	1.1

Table 4.1: Number of k-sized chunks in the training corpus

4.4.3 Evaluation

We evaluate the performance of the proposed network on MWE tagging using the three metrics described in Seddah et al. [2013], reporting for each of them the recall, precision and F-score. MWE correspond to the full MWEs, in which a predicted MWE counts as correct if it has the correct span (same group as in the gold data). MWE+POS is defined in the same fashion, except that a predicted MWE counts as correct if it has both correct span and correct POS tag. COMP correspond to the non-head components of MWEs: a non-head component of MWE counts as correct if it is attached to the head of the MWE, with the specific label that indicates that it is part of an MWE.

4.4.4 Baseline models

We compare the proposed model to our implementation of the BIOES-based model described in Collobert et al. [2011], applied to MWE tagging. We also report the results of the LIGM-Alpage architecture which obtained the best results for French SPMRL 2013 MWE recognition shared task [Constant et al., 2013]. Their system is based on Conditional Random Fields (CRF) and on external lexicons which are known to greatly improve MWE segmentation (Constant and Tellier, 2012).

4.4.5 Setup

The network is trained using stochastic gradient descent over the training data, until convergence on the validation set. Hyper-parameters are tuned on the validation set. The look-up table size for the words is 64. Word embeddings are pre-trained by performing a simple PCA on the matrix of word co-occurrences [Lebret and Collobert, 2014], using Wikipedia data. These embeddings are fine-tuned during the training process. As additional features, we only use the part-of-speech tags obtained using the freely available tool MarMoT [Mueller et al., 2013]¹. The POS-tag embedding size is 32. The context size is $c = 2$. The maximum size for a window is $K = 7$. The common embedding size for the k -window is $n_{hu} = 300$. We fix the learning rate to 0.01. Following Legrand and Collobert [2015], to prevent units from co-adapting, we adopt a dropout regularization strategy [Hinton et al., 2012] after every lookup-table, as the capacity of our network mainly lies on the input embeddings.

For the BIOES-based model, we use the following parameters: the context size is 2, word and tags feature sizes are 64 and 32 respectively, the hidden layer size is 300, the learning rate is 0.001. We use the same dropout regularization and the same word initialization as for the proposed model.

4.5 Results

We first compare our approach with the BIOES-model from Collobert et al. [2011]. Table 4.2 presents the results obtained for the two models. We see that, our model performs on par with the BIOES-based model. Interestingly, adding the POS features has little effect on the performance for MWE identification but helps to determine the MWE POS-tags.

In Table 4.3, we compare our model with the winner of the SPMRL 2013 shared task for MWE recognition [Constant et al., 2013]. Both the BIOES and chunk based models are obtained using an ensemble of 5 models, averaging the scores obtained. We see that both our model and the BIOES-based model outperform this state-of-the-art model.

¹The tags used are available here: <http://cistern.cis.lmu.de/marmot/models/CURRENT/>

Chapter 4. Phrase Prediction: a Chunk-based Approach

	COMP	MWE	MWE+POS
BIOES-model	79.4	78.5	75.4
+ WI	80.8	80.1	76.7
+ WI + POS	80.8	80.1	77.6
Chunk-model	79.1	78.3	75.2
+ WI	80.7	79.6	76.4
+ WI + POS	80.9	79.8	77.5

Table 4.2: Results on the test corpus (4043 MWEs) in terms of F-measure. WI stands for word initialization.

	COMP	MWE	MWE+POS
LIGM-Alpage	81.3	80.7	77.5
BIOES-model	81.4	80.7	78.2
Chunk-model	81.3	80.7	78.1

Table 4.3: Results on the test corpus (4043 MWEs) in terms of F-measure

4.6 Representation analysis

As the proposed chunk-based model produces continuous phrase representations, it allows for phrase comparison. Table 4.4 presents some of the closest neighbors (in terms of Euclidean distance) for some chosen phrases. We see that close representations correspond to semantically close phrases.

président de la république
chef de l'état
présidence de la république
ministre de l'intérieur
évasion fiscale
fraude fiscale
détournements financiers
libéralisme sauvage
impôt sur le revenu
impôt sur la fortune
impôt sur le patrimoine
impôts sur la fortune

Table 4.4: Closest neighbors for three input phrases in terms of Euclidean distance.

4.7 Conclusion

In this chapter, we proposed a neural network model that learns fixed-size continuous representations of arbitrarily-sized chunks by composing word embeddings. These representations are used to directly identify and classify phrases. After evaluation on the task of multiword expression tagging, our model performed on par with a baseline BIOES-based system. Fur-

thermore, we showed that it outperforms the best performing model for this task published to this date. The proposed approach does not use any external lexicon and relies on few input features. As it computes phrase representations, it allows for direct comparison between phrases. This composition procedure will be further exploited in Chapter 6.

5 Sequence Processing for Structural Inference: Syntactic Parsing

In this chapter, a word sequence modeling approach based on neural networks is proposed to tackle the problem of syntactic parsing. As opposed to the tagging task performed in the previous chapters, structural inference is required in order to solve the parsing task. In this work, parsing is cast as a greedy succession of phrase prediction problems, the latter being already addressed in the previous chapters.

The chapter is organized as follows: we first introduce the task of syntactic parsing and review the existing methods for this task. We later propose a greedy approach to parsing using word embeddings. Finally, we provide an experimental comparison with existing methods as well as an analysis of our system.

5.1 Introduction

In natural language processing (NLP), the constituency parsing task aims at analyzing the underlying syntactic structure of a natural language sequence of words, i.e. a sentence. As illustrated in Figure 5.1 the analysis is expressed as a tree of syntactic relations between sub-constituents of the sentence. In the linguistic world, Chomsky [1956] first introduced formally the parsing task, by defining the natural language syntax as a set of context-free grammar rules, i.e. a particular type of formal grammar, combined with transformations rules. Automated syntactic parsing became rapidly a key task in computational linguistics. A parse tree carries not only syntax information but might also embed some semantic information, in the sense that it can disambiguate different interpretations of a given sentence. In that respect, parsing has been widely used as an input feature for several other NLP tasks such as machine translation [Zollmann and Venugopal, 2006], information retrieval [Alonso et al., 2002], or semantic role labeling [Punyakanok et al., 2008].

In this chapter, we propose a greedy and purely discriminative parsing approach. In contrast with most existing methods, it relies on few simple features. The core of our architecture is a simple neural network which is fed with continuous word vector representations (as in

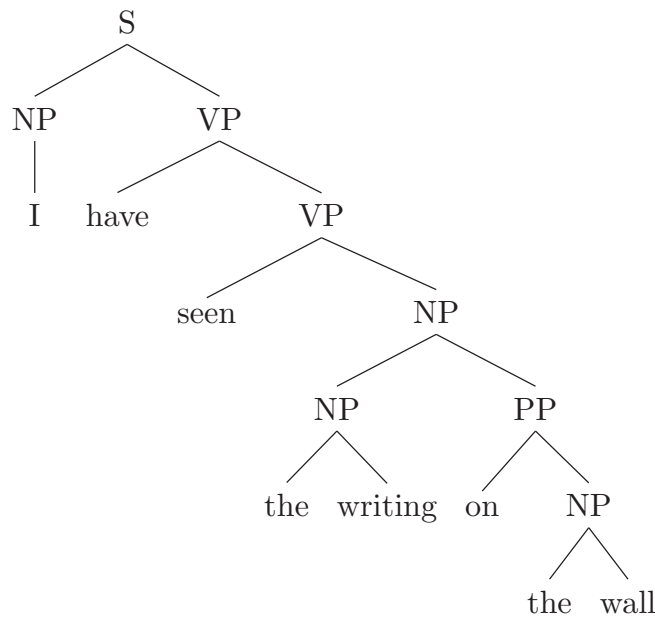


Figure 5.1: Example on syntactic parse tree.

Collobert and Weston [2008] and Socher et al. [2013a]). It models the conditional distributions of *context-aware* syntactic rules. The learned distribution rules are naturally smoothed due to the continuous nature of the input features.

5.2 Related work

The first attempts to automatically parse natural language were mainly conducted using generative models. A wide range of parser were, and still are, based on probabilistic context-free grammar (PCFGs) [Magerman, 1995, Collins, 2003, Charniak, 2000]. These types of parsers model the syntactic grammar by computing statistics of simple grammar rules (over parsing tags) occurring in a training corpus. However, many language ambiguities cannot be caught with simple tag-based PCFG rules. A key element in the success of PCFGs is to refine the rules with a word lexicon. This is usually achieved by attaching to PCFGs a lexical information called the *head-word*. Several head-word variants exist, but they all rely on a deterministic procedure which leverages clever linguistic knowledge. Parsing inference is mostly achieved using simple bottom-up chart parser [Kasami, 1965, Earley, 1970, Kay, 1986]. These methods face a classical learning dilemma: on one hand PCFG rules have to be refined enough to avoid any ambiguities in the prediction. On the other hand, too much refinement in these rules implies lower occurrences in the training set and thus a possible generalization issue. PCFGs-based parsers are thus judiciously composed with carefully chosen PCFG rules and clever regularization tricks.

5.2.1 State-Of-The-Art

Discriminative approaches from Henderson [2004], Charniak and Johnson [2005] outperform standard PCFG-based generative parsers, but only by discriminatively *re-ranking* the K -best predicted trees coming out of a generative parser. To our knowledge, the state of the art in syntactic parsing is still held by McClosky et al. [2006], who leverages discriminative re-ranking, as well as self-training over unlabeled corpora: a re-ranker is trained over a generative model which is then used to label the unlabeled dataset. The original parser is then re-trained with this new “labeled” corpus. Petrov and Klein [2007] introduced a method to automatically refine PCFG rules by iteratively splitting them. This method leverages an efficient coarse-to-fine procedure to speed up the decoding process. More recently, Finkel et al. [2008], Petrov and Klein [2008] proposed PCFG-based *discriminative* parsers reaching the performance of their generative counterparts. Conditional Random Fields (CRFs) are at the core of such approaches. Carreras et al. [2008] currently holds the state-of-the-art among the (non-reranking) discriminative parsers. Their parser leverages a global-linear model (instead of a CRF) with PCFGs, together with various new advanced features. Huang et al. [2010] showed that jointly using multiple self-trained grammars can achieve higher accuracy than an individual grammar.

In contrast to these existing approaches, our parser does not rely on PCFGs, nor on refined features like head-words. Tagging nodes is achieved in a greedy manner, using only raw words and part-of-speech (POS) as features.

5.2.2 Greedy Parsing

Many discriminative parsers follows a greedy strategy because of the lack (or the intractability) of a global tree score for an entire derivation path which would combine independent node decisions. Adopting a greedy strategy that maximize local scores for individual decisions is then a solution worth investigating. One of the first successful discriminative parsers [Ratnaparkhi, 1999] was based on maximum entropy classifiers (trained over a large number of different features) and powered a greedy shift-reduce strategy. Henderson [2003] introduced a generative left-corner parser where the probability of a derivation given the derivation historic was approximated using a simple synchrony networks, which is a neural network specifically designed for processing structures. Turian and Melamed [2006] later proposed a bottom-up greedy algorithm following a left-to-right or a right-to left strategy and using a feature boosting approach. In this approach, greedy decisions regarding the tree construction are made using decision tree classifiers. Their model was nevertheless limited to short length sentences. Zhu et al. [2013] proposed a shift-reduce parser which achieves results comparable to their chart-based counterparts. This is done by leveraging several unsupervised trained features (word Brown clustering, dependency relations, dependency language model) combined with a smart beam search strategy.

5.2.3 Syntactic parsing using word embeddings

Several works leveraging continuous vector representations have been previously proposed for syntactic parsing. Collobert [2011] introduced a neural network-based approach, iteratively tagging “levels” of the parse tree where the full sentence was seen at each level. A complex pooling approach was introduced to capture long-range dependencies, and performance only matched early lexicalized parsers. Socher et al. [2011b] introduced a recursive approach, where representations are “compressed” two by two to form higher-level representations. However, the system was limited to bracketing, and did not produce parsing tags. The authors later proposed an improved version in Socher et al. [2013a], where their approach was used to re-rank the output of the Stanford Parser, approximately reaching state-of-the-art performance. In contrast, our approach does not re-rank an external generative parser.

5.3 A greedy discriminative parser

5.3.1 Smoothed Context Rule Learning

PCFG-based parsers rely on the statistical modeling of rules of the form $A \rightarrow B, C$, where A, B and C are tree nodes. The context-free grammar is always normalized in the Chomsky normal form (CNF) to make the global tree inference practical (with a dynamic programming algorithm like CYK or similar). In general a tree node is represented as several features, including for example its own parsing tags and head word (for non-terminal nodes) or word and Part Of Speech (POS) tag (for terminal nodes) [Collins, 2003]. State-of-the-art parsers rely on a judicious blending of carefully chosen features and regularization: adding features in PCFG rules might resolve some ambiguities, but at the cost of sparser occurrences of those rules. In that respect, the learned distributions must be carefully smoothed so that the model can generalize on unseen data. Some parsers also leverage other types of features (such as bigram or trigram dependencies between words [Carreras et al., 2008]) to capture additional regularities in the data.

In contrast, our system models non-CNF rules of the form $A \rightarrow B_1, \dots, B_K$. The score of each rule is determined by looking at a large context of tree nodes. More formally, we learn a classifier of the form:

$$f(C_{left}, B_1, \dots, B_K, C_{right}) = (s_1, \dots, s_{|\mathcal{T}|}) \quad (5.1)$$

where the B_k are either terminal or non-terminal nodes, K is the size of the right part of the rule, C_{left} and C_{right} are context terminals or non-terminals and s_t is the score for the parsing tag $t \in \mathcal{T}$. Each possible rule $A_i \rightarrow B_1, \dots, B_K$ is thus assigned a score s_i by the classifier (with $A_i \in \mathcal{T}$). These scores can be interpreted as probabilities by performing a softmax operation. We used a Multi Layer Perceptron (MLP) as classifier. Formal details are presented in Section 5.4.2.

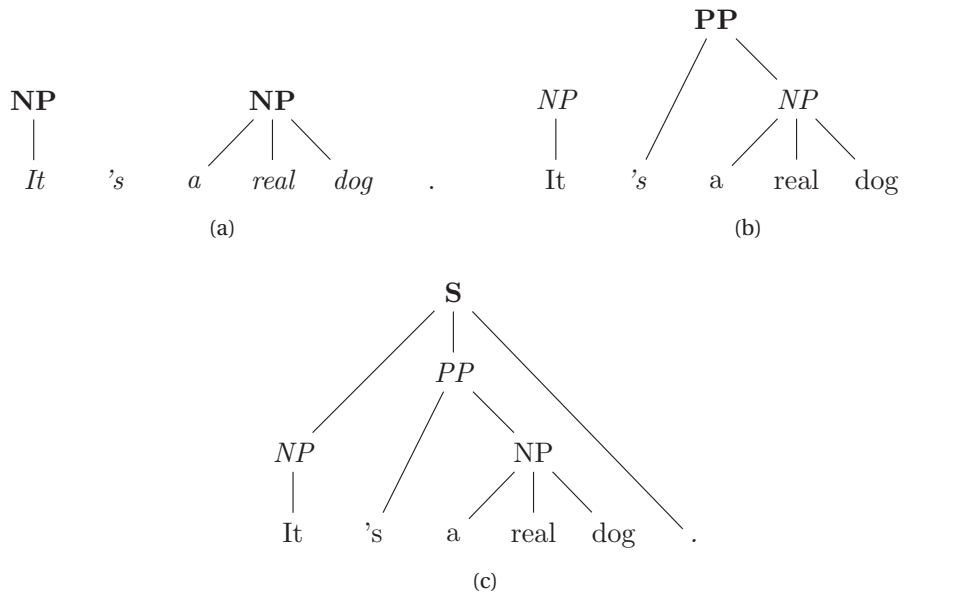


Figure 5.2: Illustration of the greedy algorithm: at each iteration (a)→(b)→(c), the classifier sees only the previous tree heads (ancestors), shown here in italics. It predicts new nodes (here in bold). New tree heads become the ancestors at the next iteration. All other previously discovered tree nodes (shown in regular black here) will remain unchanged and ignored in subsequent iterations.

The only tree node features considered in our system are parsing tags (or POS tags for terminals), as well as the headword (or words for terminals). We overcome the problem of data sparsity which occurs in most classical parsers by leveraging *continuous vector representations* for all features associated to each tree node. In particular, word (or headword) representations are derived from recent distributed representations computed on large unlabeled corpora (such as Collobert and Weston [2008], Dhillon et al. [2011]). For instance, thanks to this approach, our system can naturally generalize a rule like $NP \rightarrow a, clever, guy$ to a possibly unseen rule like $NP \rightarrow a, smart, guy$, as the vector representation of *smart* and *clever* are close to each other, given that they are semantically and syntactically related.

5.3.2 Greedy Recurrent Algorithm

Our parser follows a bottom-up iterative approach: the tree is constructed starting from the terminal nodes (sentence words). Assuming that a part of the tree has been already predicted (see Figure 5.2), the next iteration of our algorithm looks for all possible new tree nodes which combine ancestors (i.e., heads of the trees predicted so far). New nodes are found by maximizing the score of our context-rule classifier (equation 5.1), constrained in such a way so that two new nodes cannot overlap, thanks to a dynamic programming approach. The system is recurrent, in the sense that new predicted parsing labels are used in the next iteration of our

algorithm.

For each iteration, assuming N ancestors

$$X = [X_1, \dots, X_N], \quad (5.2)$$

finding all possible new nodes with K ancestors would require to apply

$$f(C_{left}, B_1, \dots, B_K, C_{right}) \quad (5.3)$$

over all possible windows of K ancestors in X . One would also have to vary K from 1 to N , to discover new nodes of all possible sizes. Obviously, this could quickly become time consuming for large sentence sizes. This problem of finding nodes with a various number of ancestors can be viewed as the classical NLP problem of finding “chunks” of various sizes. This problem is typically transformed into a tagging task: finding the chunk with label A in the rule $A \rightarrow X_i, X_{i+1}, \dots, X_j$ can simply be viewed as tagging the ancestors with $B-A, I-A, \dots, E-A$, where we use the standard BIOES label prefixing (*Begin, Intermediate, Other, End, Single*). See Table 5.1 for a concrete example. The classifier outputs the “*Other*” tag, when the considered ancestors do not correspond to any possible rule.

In the end, our approach can be summarized as the following iterative algorithm:

1. Apply a sliding window over the current ancestors: the neural network classifier (5.1) is applied over all K consecutive ancestors X_1, \dots, X_N , where K has to be carefully tuned.
2. Aggregate BIOES tags into chunks: a dynamic program (based on a CRE, as detailed in Section 5.4.3) finds the most likely sequence of BIOES parsing tags. The new nodes are then constructed by simply aggregating BIES tags

$$B-A, I-A, \dots, E-A$$

into A (for any label A).

3. Ancestors tagged as O , as well as newly found tree nodes are passed as ancestors to the next iteration.

The tree construction ends when there is only one ancestor remaining, or when the classifier did not find any new possible rule (everything is tagged as O).

5.4 Architecture

In this section, we formally introduce the classification architecture used to find new tree nodes at each iteration of our greedy recurrent approach. A simple two-layer neural network is at the core of the system. It leverages continuous vector word representations. In this respect,

Table 5.1: A simple example of a grammar rule extracted from the sentence “*It’s a real dog.*”, and its corresponding BIOES grammar. In both cases, we include a left and right context of size 1. The middle column shows the required classifier evaluations. The right column shows the type of scores produced by the classifier.

GRAMMAR	CLASSIFIER EVALUATIONS	SCORES
NP → 's A REAL DOG .	$f('s, A, REAL, DOG, .)$	$s_{NP}, \dots, s_{VP}, s_O$
B-NP → 's A REAL	$f('s, A, REAL)$	$s_{B-NP}, \dots, s_{E-VP}, s_O$
I-NP → A REAL DOG	$f(A, REAL, DOG)$	
E-NP → REAL DOG .	$f(REAL, DOG, .)$	

the network is clearly inspired by the work of Bengio et al. [2001] in the context of language modeling, and later re-introduced in Collobert et al. [2011] for various NLP tagging tasks.

Given an input sequence of N tree node ancestors X_1, \dots, X_N (as defined in Section 5.3.2), our model outputs a BIOES-prefixed parsing tag for each ancestor X_i , by applying a sliding window approach. These scores are then fed as input to a properly constrained graph on which we apply the Viterbi algorithm to infer the best sequence of parsing tags. The whole architecture (including transition scores in the graph) is trained in an end-to-end manner by maximizing the graph likelihood. The system can be viewed as a particular Graph Transformer Network [Bottou et al., 1997], or a particular *non-linear* Conditional Random Field (CRF) for sequences [Lafferty et al., 2001]. Each layer of the architecture is presented in detail in the following paragraphs. The objective function will be introduced in Section 5.4.4.

5.4.1 Words Embeddings

Given a sentence of N words, w_1, w_2, \dots, w_N , each word $w_n \in \mathcal{W}$ is first embedded in a D -dimensional vector space by applying the lookup-table operation described in Section 3.4.1:

$$LT_{\mathbf{W}}(w_n) = \mathbf{W}_{w_n}, \quad (5.4)$$

where the matrix $\mathbf{W} \in \mathbb{R}^{D \times |\mathcal{W}|}$ represents the parameters to be trained in this lookup layer. Each column $\mathbf{W}_n \in \mathbb{R}^D$ corresponds to the vector embedding of the n^{th} word in our dictionary \mathcal{W} .

In practice, it is common to give several features (for each tree node) as input to the network. This can be easily done by adding a different lookup table for each discrete feature. The input becomes the concatenation of the outputs of all these lookup-tables:

$$\begin{aligned} LT_{W_1, \dots, W_k}(w_n) &= (LT_{W_1}(w_n))^T, \\ &\dots, \\ &(LT_{W_k}(w_n))^T \end{aligned} \quad (5.5)$$

where $|\mathcal{F}|$ is the number of features. For simplicity, we consider only one lookup-table in the rest of the architecture description.

5.4.2 Sliding Window BIOES Tagger

The second module of our architecture is a simple neural network which applies a sliding window over the output of the lookup tables, as shown in Figure 5.3. The n^{th} window is defined as

$$u_n = [LT(X_{n-\frac{K-1}{2}}), \dots, LT(X_n), \dots, LT(X_{n+\frac{K-1}{2}})], \quad (5.6)$$

where K is the size of window. The module outputs a vector of scores $\mathbf{s}(u_n) = [s_1, \dots, s_{|\mathcal{T}|}]$ (where s_t is the score of the BIOES-prefixed parsing tag $t \in \mathcal{T}$ for the ancestor X_n). The ancestors with indices exceeding the input boundaries ($n - (K - 1)/2 < 1$ or $n + (K - 1)/2 > N$) are mapped to a special padding vector (which is also learned). As any classical two-layer neural network, our architecture performs several matrix-vector operations on its inputs, interleaved with some non-linear transfer function $h(\cdot)$,

$$s(u_n) = M_2 h(M_1 u_n), \quad (5.7)$$

where the matrices $M_1 \in \mathbb{R}^{H \times K|D|}$ and $M_2 \in \mathbb{R}^{|\mathcal{T}| \times H}$ are the trained parameters of the network. The number of hidden units H is a hyper-parameter to be tuned.

As transfer function, we chose in our experiments a (fast) “hard” version of the hyperbolic tangent:

$$h(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases} \quad (5.8)$$

5.4.3 Aggregating BIOES Predictions

The scores obtained from the previous module of our architecture are in BIOES format. The next module in our system aggregates these tags and finds the new tree nodes at each iteration of our greedy recurrent approach. We introduce a graph G of scores as shown in Figure 5.4: each node of the graph corresponds to a BIOES score produced for each ancestor by the neural network module. This graph is constrained in such a way that only feasible sequences of tags are possible (e.g. $B-A$ tags can only be followed by $I-A$ tags, for any parsing label A). Our graph also includes a duration model: on each edge, we add a transition score $A_{t,t'}$ for jumping from tag $t \in \mathcal{T}$ to $t' \in \mathcal{T}$.

A score for a sequence of tags $[t]_1^N$ in the lattice G is obtained as the sum of scores along $[t]_1^N$

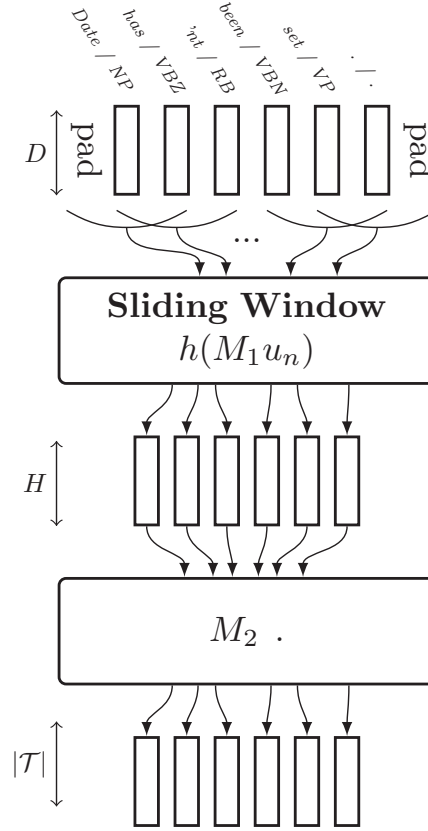


Figure 5.3: Sliding window tagger. Given the concatenated output of lookup tables (here the ancestor words/headwords and ancestor tags), the tagger outputs a BIOES-preixed parsing tag for each ancestor node. The neural network itself is a standard two-layer neural network.

in G :

$$S([t]_1^N, [u]_1^N, \theta) = \sum_{n=1}^N (A_{t_{n-1}t_n} + s(u_n)_{t_n}), \quad (5.9)$$

where θ represents all the trainable parameters of the complete architecture. The sequence of tags $[t^*]_1^N$ for the input sequence of tree node ancestors X_1, \dots, X_N is then inferred by finding the path which leads to the maximal score:

$$[t^*]_1^N = \operatorname{argmax}_{[t]_1^N \in \mathcal{T}^N} S([t]_1^N, [u]_1^N, \theta) \quad (5.10)$$

The Viterbi algorithm is the natural choice for this inference. From this optimal BIOES tag sequence, we extract sub-sequences $B-A, \dots, E-A$ and $S-A$ as new nodes for the tree. O tags are simply ignored. See Section 5.3.2 for more details.

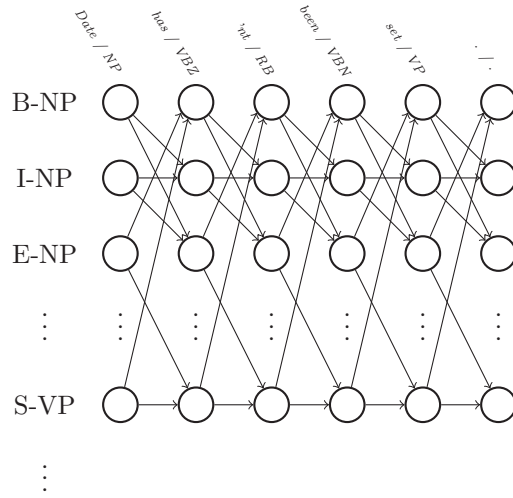


Figure 5.4: Constrained graph for tag inference. Only feasible sequences of tags are considered. The nodes of the graph are assigned a score from the tagger shown in Figure 5.3. Edges of the graph are assigned a transition score which is learned similarly to other parameters in the architecture.

5.4.4 Training Likelihood

Our architecture sees sequences of ancestor tree nodes, and outputs new possible syntactic tree nodes only from this history. Technically speaking, the training set can be prepared by iterating over each tree in the training corpus, removing all possible leaves in an iterative process so that all training rules are uncovered (see Figure 5.5). It implies that the system is only trained on *correct* sequences of tree nodes. In that respect, it is not trained to recover from past mistakes it could have made during the recurrent process.

The neural network is trained by maximizing a likelihood over the training data, using stochastic gradient descent. The score for a path can be interpreted as a conditional probability over this path by exponentiating score (thus making it positive) and normalizing it with respect to all possible paths. We define \mathcal{P} as the set of possible tag paths in the constrained graph G , as shown in Figure 5.4. The log-probability of a sequence of tags $[t]_1^N$ given the lookup table representations $[u]_1^N$ is given by:

$$\log P([t]_1^N | [u]_1^N, \theta) = S([t]_1^N, [u]_1^N, \theta) - \text{logadd}_{\forall [t']_1^N \in \mathcal{P}} S([t']_1^N, [u]_1^N, \theta) \quad (5.11)$$

where we adopt the notation $\text{logadd}_{z_n} = \log(\sum_i e^{z_i})$.

Computing the log-likelihood efficiently is not straightforward, as the number of terms in the logadd grows exponentially with the length of the sentence. Fortunately, it can be computed in linear time with the Forward algorithm, which derives a recursion similar to the Viterbi al-

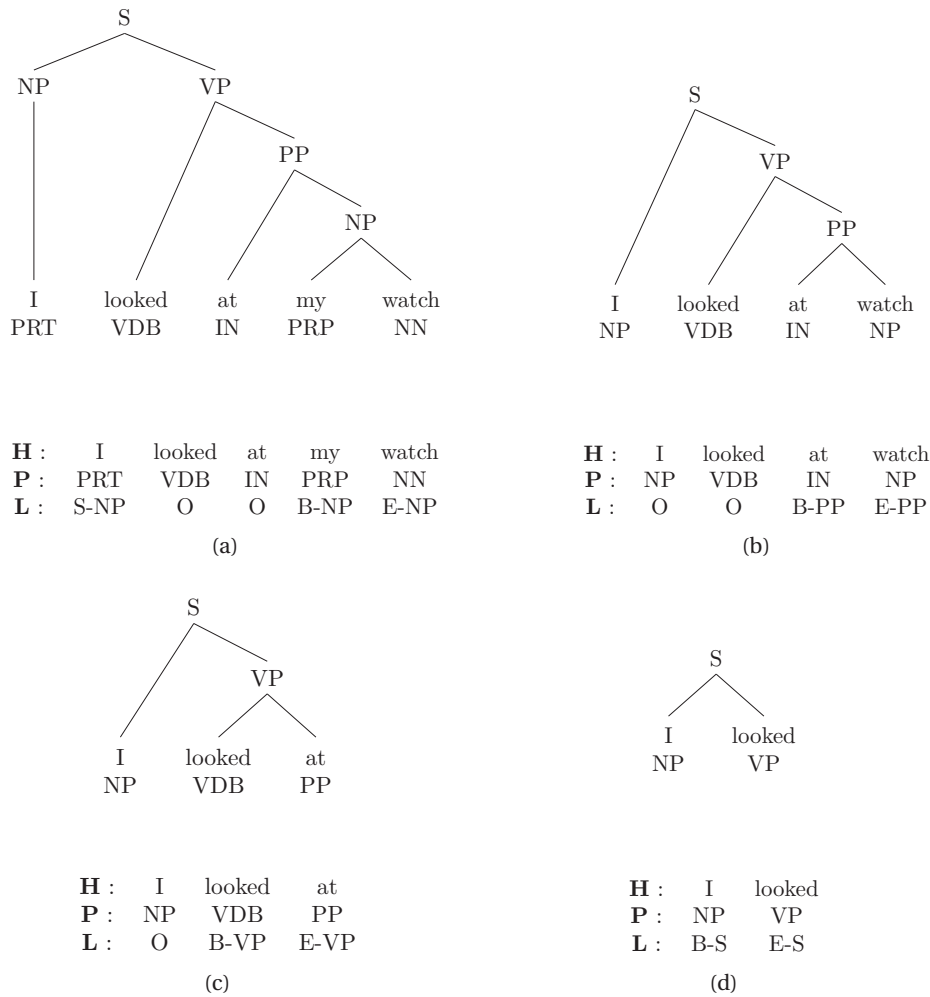


Figure 5.5: Iterative procedure (a)→(b)→(c)→(d) to generate the training data, which involves cutting out all tree leaves at each step. The data fed to our network architecture is then easily uncovered (H: ancestor headwords/words, P: ancestor POS/parsing tags, L: parsing labels to be predicted).

gorithm (see Rabiner [1990]). The complete architecture is trained by simply backpropagating through this recursion, up to the lookup layers. Note that the likelihood (5.11) corresponds to a standard CRF model for sequences.

5.5 Experiments

5.5.1 Corpus

Experiments were performed using the standard English Penn Treebank data set (Marcus et al., 1993). We used the classical parsing setup, with sections 02-21 used to train our model, section 22 used as validation for choosing all our hyper-parameters, and section 23 used for testing. We applied only a small subset of the typical pre-processing set over the data: (1) functional labels, traces were removed, (2) the PRT label was replaced as ADVP [Magerman, 1995].

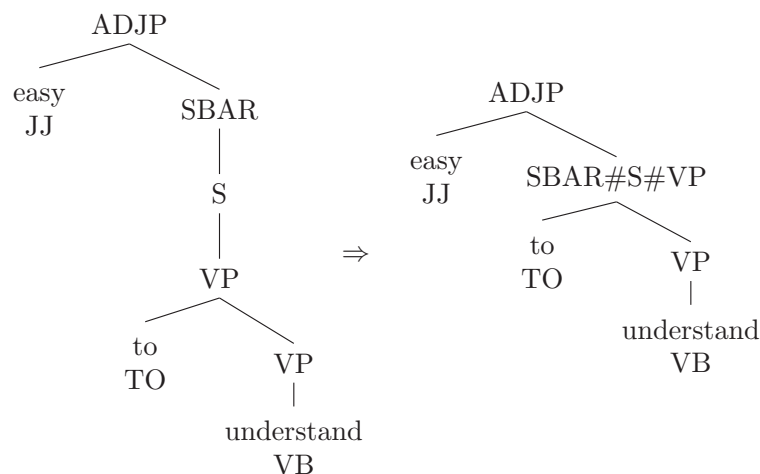


Figure 5.6: Training corpus pre-processing. Original Penn Treebank trees containing non-terminal nodes with only one non-terminal node (left), and after concatenating those nodes (right).

The Penn Treebank data set contains non-terminal tree nodes which only have one non-terminal child, as shown in Figure 5.6. To avoid possible looping issues (called unary chain issue in the literature) in our parsing algorithm (e.g. a node being repetitively tagged with two different tags in our iterative process), we transformed the *training* corpus so that non-terminal nodes having only one non-terminal child were merged together, and take as tag the concatenation of all merged node tags (see Figure 5.6). This way, the system learns that a node must contain at least two ancestors. The iterative process is thus guaranteed to converge. We kept only concatenated labels which occurred at least 30 times (corresponding to the lowest number of occurrences of the less common original parsing tag), leading to 11 additional parsing tags. Added to the original 26 parsing tags, this resulted in 149 tags produced by our parser (148 BIOES-prefixed tags + Other). At test time, the inverse operation is performed: concatenated tag nodes are simply expanded into their original form.

5.5.2 Features

We consider the following features to train our architecture:

- Words and headwords:
 - For terminal nodes, the word itself, in low caps¹. As in Collins [2003], words occurring 5 times or less were mapped to an “UNKNOWN” word.
 - For non-terminal nodes: headwords, following the procedure described in Collins [2003].
- POS tags (for terminals) or parsing tags of the node’s ancestors (for non-terminals). POS tags were produced with SENNA [Collobert et al., 2011].
- POS tags of headwords.

5.5.3 Setup

We train the network using stochastic gradient descent over the available training data, until convergence on the validation set. We select the following hyper-parameters according to the validation. Lookup-table sizes for the words and tags (part-of-speech and parsing) are 100 and 20, respectively. The window size for the tagger is $K = 7$ (3 neighbors from each side). The size of the tagger’s hidden layer is $H = 500$. We used the word embeddings obtained from Lebrecht and Collobert [2014] to initialize the word lookup-table. These embeddings were then fine-tuned during the training process. Finally, we fixed the learning rate to $\lambda = 0.025$ during the stochastic gradient procedure. The only “trick” used during training was to divide the learning rate by the input size of each linear layer [Plaut and Hinton, 1987].

5.5.4 Results

Table 5.2 shows the importance of the different features we used. Even though the training procedure is non-convex, the variance of the F1 score over 20 different runs (for the architecture Word + POS + hw + wi) was only 0.01.

Since our architecture performs the decoding very quickly, we additionally performed a voting procedure using several models learned from different random initializations. We averaged all neural network classifiers (ignoring their own respective CRF decoding part) and trained a new CRF on top of it (without fine-tuning any of the neural network classifiers). The scores obtained with 10 classifiers are shown in Table 5.3.

Results in Table 5.3 are reported in terms of recall (R), precision (P) and F1 score. Scores were

¹Adding a capital feature had no impact on the performance of our parser. Note that POS tags were generated with the original caps in the sentence.

FEATURE	F1
WORD + POS	85.1
WORD + POS + HW	86.9
WORD + POS + WI	86.2
WORD + POS + HW + WI	88.3

Table 5.2: Influence of different features. Results are given in terms of F1-score. POS = part-of-speech, hw = head-word, wi = word initialization from Lebrete and Collobert [2014].

	MODEL	(R)	(P)	F1	(R)	(P)	F1
GENERATIVE	MAGERMAN (1995)	84.6	84.9	84.8			
	COLLINS (1999)	88.5	88.7	88.6	88.1	88.3	88.2
	CHARNIAK (2000)	90.1	90.1	90.1	89.6	89.5	89.6
GENERATIVE WITH RE-RANKING	HENDERSON (2004)				89.8	90.4	90.1
	CHARNIAK AND JOHNSON (2005)			92.0			91.1
	SOCHER ET AL (2013)			91.1			90.4
	MCCLOSKEY ET AL (2006)						92.1
PURELY DISCRIMINATIVE	PETROV AND KLEIN (2008)			90.0			89.4
	CARRERAS ET AL. (2008)				90.7	91.4	91.1
	OUR MODEL	88.4	89.0	88.7	88.0	88.6	88.3
	OUR MODEL (VOTING)	90.0	90.1	90.1	89.6	89.7	89.6

Table 5.3: Results in terms of Precision (P), Recall (R), and F1 score. The reported time is the time to parse the full WSJ test corpus.

obtained using the Evalb implementation². We compare our system with several other parsers. We chose to report the scores of the three main generative parsers, as well as those of known re-ranking parsers. We also considered two major purely discriminative parsers.

5.5.5 Rule Prediction Analysis

Figure 5.7 shows the output of the classifier (applied on every possible window of size 7) for the sentence "When the little guy gets frightened, the big guys hurt badly.". For this sentence, the expected rule are the following:

WHADVP → When
 NP → the little guy
 ADJP → frightened
 NP → the big guys
 ADVP → badly

It is interesting to see that the network alone is able to predict all the rules of the sentence. The CRF is however essential to produce a consistent output, by aggregating BIES prefixed chunks.

²Available at <http://nlp.cs.nyu.edu/evalb/>

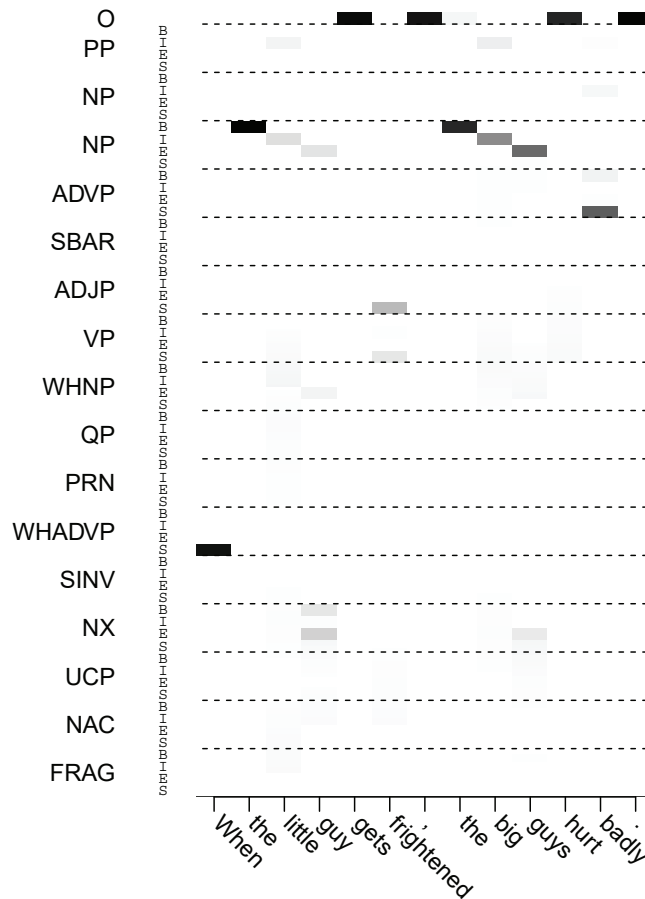


Figure 5.7: Normalized scores from the network classifier (black means high score) for the sentence "When the little guy gets frightened, the big guys hurt badly.". Each tag is in BIOES form (y axis). Each ancestor in the input is on the x axis.

5.6 Conclusion

We presented a simple model that is able to learn syntactic grammar rules surprisingly well, considering the simple features employed. This parser achieves performance approaching those obtained by state-of-the-art re-ranking systems and it performs almost the best amongst purely discriminative parsers. Due to its simplicity, there are many possibilities for further improvement. In particular, the use of the head-word procedure [Collins, 2003], selecting only the most relevant word in a sub-tree, should be revisited.

In the next chapter, the proposed model is enhanced using a compositional procedure that learns a higher-level sub-tree representation, in the spirit of Socher et al. [2013a].

6 RNN-Based Phrase Composition for Syntactic Parsing

In the previous chapter, we introduced a greedy approach to syntactic parsing in which the task is cast as multiple phrase prediction problems that are solved recursively in a greedy manner. In this chapter, this model is enhanced using the composition operation introduced in Chapter 3. This procedure is recursively applied in order to perform a syntactic and semantic summary of the contents of sub-trees in the form of fixed-size vector representations. Both the composition and node prediction are trained *jointly*.

The chapter is organized as follows: We first present several related approaches making use of recursive neural networks for syntactic parsing. We then introduce our novel composition procedure. We finally provide an empirical evaluation of our models as well as an analysis of our compositional vectors.

6.1 Related Work

RNN were seen very early [Elman, 1991] as a way to tackle the problem of parsing, as they can naturally recur along the parse tree. A first practical application of RNN on syntactic parsing was proposed by Costa et al. [2002]. Their approach was based on a left-to-right incremental parser, where a recursive neural network was used to re-rank possible phrase attachments. The goal of their contribution was, in their own terms, the assessment of a methodology rather than a fully functional system. They demonstrated that RNNs were able to capture enough information to make correct parsing decisions.

Collobert [2011] proposed a purely discriminative parser based on neural networks. This model leveraged continuous vector representations from Collobert and Weston [2008], and builds the full parsing tree in a bottom-up manner. To deal with the recursive structure inherent to syntactic parsing, a very simple history was given to the network as a new vector feature (corresponding to the nearest tag spanning the word being tagged).

Socher et al. [2011a] also leveraged continuous vectors from Collobert and Weston [2008], combining them to build a tree in a greedy manner. However, this work did not tackle the full

parse tree problem, but was restricted to unlabeled bracketing. Socher et al. [2013a] introduced the compositional vector grammar (CVG) which combines PCFGs with a syntactically untied recursive neural network (SU-RNN). Composition is performed over a binary tree, then used to score the K -best trees coming out of a generative parser. For a given (parent) node of the tree, the authors apply a composition operation over its child nodes, conditioned with their syntax information. In contrast, we compose phrases (not limited to two words). Both the words and syntax information of the child nodes are fed to each composition operation, leading to a vector representation of each tree node carrying both some semantic and syntactic information. We also do not rely on any generative parser as our model *jointly* trains the task of node prediction, and the task of node composition.

Chen and Manning [2014] proposed a greedy transition-based dependency parser based on neural networks, fed with dense word and tag vector representations. In contrast to our approach, it does not integrate a compositional procedure over sentence sub-trees. The network is only involved in predicting correct transitions at each step of the parsing process.

6.2 Greedy RNN Parsing

As introduced in Chapter 5, our parser is based on a neural network tagger, and performs parsing in a greedy recurrent way. Our approach is a bottom-up iterative procedure: the tree is constructed starting from the terminal nodes (sentence words), as shown in Figure 6.1. Including the new composition procedure, our procedure can be summarized as the following iterative algorithm. Note that the novelty resides in step 3:

1. We look for all possible new tree nodes merging input constituents (i.e., heads of the trees predicted so far or leaves which have not been composed so far). For that purpose, we apply a neural network sliding window tagger (as described in Section 5.4.2) over input constituents X_1, \dots, X_N . Considering an arbitrary rule

$$A \rightarrow X_i, X_{i+1}, \dots, X_j$$

defining a new node with tag A , the tagger will produce prefixed tags $B-A, I-A, \dots, E-A$, respectively for constituents X_i, X_{i+1}, \dots, X_j , following a classical BIOES prefixing scheme

2. A simple dynamic programming (as described in Section 5.4.3) is performed, only to insure the coherence of the tag prediction (e.g., a $B-A$ can be followed only by a $I-A$ or a $E-A$).
3. A neural network composition module computes vector representations of the new nodes, according to the representations of the merged constituents, as well as the tag predictions (see Figure 6.2).
4. New predicted nodes become input constituents and we go back to 1 (see Figure 5.2).

\mathbf{I}_W :	Look	around	and	choose	your	own	ground	.
\mathbf{I}_T :	VB	RP	CC	VB	PRP\$	JJ	NN	.
\mathbf{O} :	O	S-PRT	O	O	B-NP	I-NP	E-NP	O
<hr/>								
\mathbf{I}_W :	Look	\mathbf{r}_1	and	choose	\mathbf{r}_2	.		
\mathbf{I}_T :	VB	PRT	CC	VB	NP	.		
\mathbf{O} :	B-VP	E-VP	O	B-VP	E-VP	O		
<hr/>								
\mathbf{I}_W :	\mathbf{r}_3	and	\mathbf{r}_4	.				
\mathbf{I}_T :	VP	CC	VP	.				
\mathbf{O} :	B-VP	I-VP	E-VP	O				
<hr/>								
\mathbf{I}_W :	\mathbf{r}_5	.						
\mathbf{I}_T :	VP	.						
\mathbf{O} :	B-S	E-S						

Figure 6.1: Greedy parsing algorithm, on the sentence “Look around and choose your own ground.”. \mathbf{I}_W , \mathbf{I}_T and \mathbf{O} stand for input words (or composed word representations \mathbf{r}_i), input syntactic tags (parsing or part-of-speech) and output tags (parsing), respectively. See Figure 6.2 and Section 6.2.1 for the word composition procedure. The tree produced after 4 greedy iterations (as shown here) can be reconstructed as the following: (S (VP (VP (VB Look) (PRT (RP around)))) (CC and) (VP (VB choose) (NP (PRP\$ your) (JJ own) (NN ground)))) (. .)).

Our system is recurrent in two ways: newly predicted parsing node labels as well as vector representations obtained by composing these predicted nodes, are used in the next iteration of our algorithm.

6.2.1 Word-Tag Composition

At each step of the parsing procedure, we represents each node of the tree as a vector representation, which summarizes both the syntax (predicted POS or parsing tags) and the semantic (words) of the sub-tree corresponding to the given node. As shown in Figure 6.2, the vector representation is obtained by a simple recurrent procedure, which involves several components:

- Word vector representations *for the leaves* (coming out from a lookup table) (dimension D).
- Tag (POS for the leaves, predicted tags otherwise) vector representations (also coming out for another lookup table, as explained in Section 5.4.1) (dimension T).
- Compositional networks $C_k()$. Each of them can compress the representation of a chunk of size k into a D -dimensional vector.

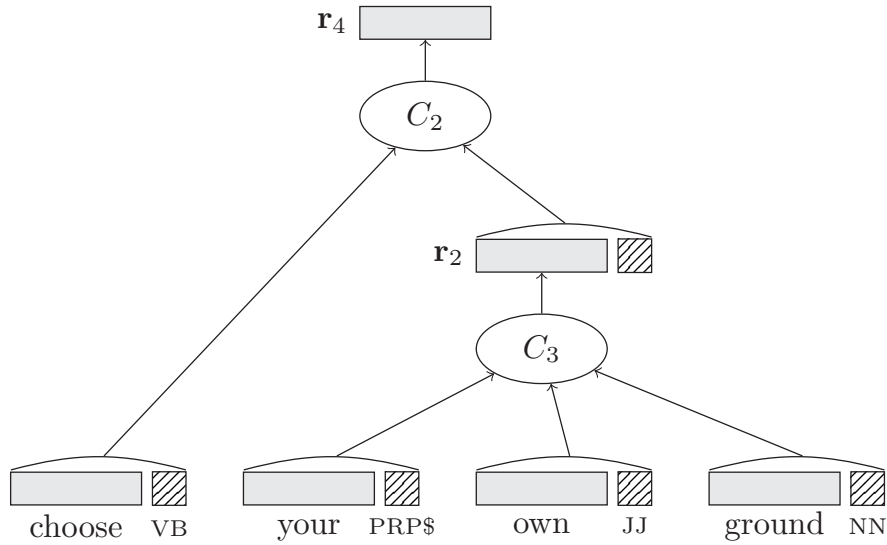


Figure 6.2: Recurrent composition of the sub-tree (VP (VB choose) (NP (PRP\$ your) (JJ own) (NN ground))). The representation \mathbf{r}_2 is first computed using the 3-inputs module C_3 with your/PRP\$ own/JJ ground/NN as input. \mathbf{r}_4 is obtained by using the 2-inputs module C_2 with choose/VB R1/NP as input

Compositional networks take as input both the merged node representations and predicted tag representations. There is one different network C_k for each possible node with a number of k merged constituent. In practice most tree nodes do not merge more than a few constituents¹. In our case, denoting $\mathbf{z} \in \mathbb{R}^{(D+T) \times k}$ the concatenation of the merged constituent representations (k vectors of tags and constituent representations), the compositional network is simply a matrix-vector operation followed by a non-linearity

$$C_k(\mathbf{z}) = h(\mathbf{M}^k \mathbf{z}), \tag{6.1}$$

where $\mathbf{M}^k \in \mathbb{R}^{D \times (k(D+T))}$ is a matrix of parameters to be trained, and $h()$ is a simple non-linearity such as a pointwise hyperbolic tangent.

Note that node and word representations are embedded in the same space. This way, the compositional networks C_k can compress indifferently information coming from leaves or sub-trees. Implementation-wise, one can store new node representations into the word lookup-table as the tree is created, such that subsequent composition or tagging operations can be achieved in an efficient manner.

¹Taking $1 \leq k \leq 5$ covers already 98.6% of the nodes in the Wall Street Journal training corpus, and $1 \leq k \leq 7$ covers 99.8%.

6.2.2 Training Procedure

As introduced in Chapter 5, both the composition network and tagging networks are trained by maximizing a likelihood over the training data using stochastic gradient ascent. As introduced in section 5.4.4, we performed all possible iterations, over all training sentences, of the greedy procedure presented in Figure 6.1 constrained with the provided labeled parse tree. This leads to our training set of sequences of tree nodes. For every tree node, the sub-trees (structure and tags) were also extracted during this procedure.

Training the system consists in repeating the following steps:

- Pick a random sequence of nodes extracted in the training set, as described above. Consider the associated sub-trees for each node which is not a leaf.
- Perform a forward pass of the word-tag composer (see Section 6.2.1) along these sub-trees.
- For all nodes in the sequence, perform a forward pass of the tagger according to word (or sub-tree) representations, as well as constituent tags.
- Compute a *likelihood of the right sequence of BIOS-prefixed tags* (as described in Section 5.4.4), given the scores of the tagger.
- Backward gradient through the tagger up to the word (or sub-tree) and tag representations.
- Backward gradient through the word-tag composer up to the word and tag representation.
- Update all model parameters (from compositional networks C_i , tagger network, and lookup tables) with a fixed learning rate.

6.3 Experiments

Experiments were conducted using the corpus and preprocessing described in Chapter 5. This section describes the setup used for our experiments and introduces the dropout regularization used to prevent overfitting. Finally, we present an experimental comparison with existing methods as well as an analysis the sub-tree representations learned by our model.

6.3.1 Detailed Setup

Our systems were trained using a stochastic gradient descent over the available training data until convergence on the validation set. Hyper-parameters were chosen according to the validation. Lookup-table sizes for the words and tags (part-of-speech and parsing) are 200 and 20, respectively. The window size for the tagger is $K = 7$ (3 neighbours from each side). The size of the tagger’s hidden layer is $H = 500$. We used the word embeddings obtained

from Lebrecht and Collobert [2014] to initialize the word lookup-table. These embeddings were then fine-tuned during the training process. We fixed the learning rate to $\lambda = 0.15$ during the stochastic gradient procedure. As suggested in Plaut and Hinton [1987], the learning rate was divided by the size of the input vector of each layer. The part-of-speech tags were obtained using the freely available software SENNA².

6.3.2 Word Embedding Dropout Regularization

We found that our system was easily subject to overfitting (training F1-score increasing while the validation curve was eventually decreasing as shown in Figure 6.3). As the capacity of our network mainly lies on the words and tag embeddings, we adopted a dropout regularization strategy [see Hinton et al., 2012] for the lookup tables. The key idea of the dropout regularization is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. In our case, during the training phase, a “dropout mask” is applied to the output of the lookup-tables: each element of the output is set to 0 with a probability 0.25. At test time, no patch is applied but the output is re-weighted, scaling it by 0.75. We observed a good improvement in F1-score performance, as shown in Figure 6.4.

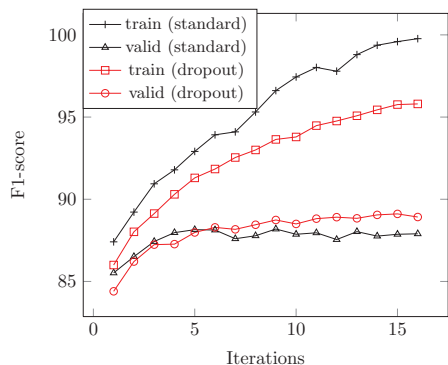


Figure 6.3: Train and validation F1-score, according to the number of training iterations, with and without the “dropout” procedure.

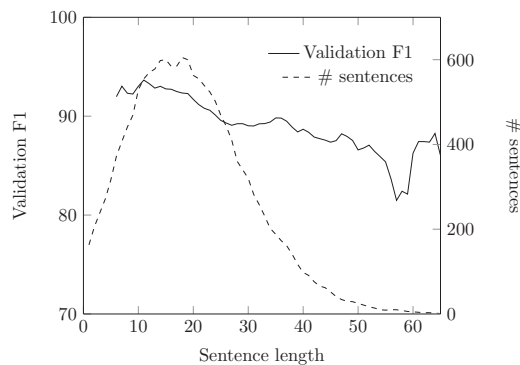


Figure 6.4: Validation F1 and number of sentences, according to the the sentence length.

6.3.3 Performance comparison

F1 performance scores are reported in Table 6.1. Scores were obtained using the Evalb implementation³. We compared our system is compared with a range of different state-of-the-art parsers. In addition to the the four main generative parsers, we report the scores of well known re-ranking parsers (including the state-of-the-art from McClosky et al. [2006]) as well as for

²<http://ml.nec-labs.com/senna>

³Available at <http://nlp.cs.nyu.edu/evalb>

two major purely discriminative parsers. Detailed error analysis compared against a subset of these parsers is reported in Table 6.2, using the code provided by Kummerfeld et al. [2012]. Performance with respect to sentence length is reported in Figure 5.

We included a voting procedure using several models trained starting from different random initializations. The voting procedure is achieved in the following way: at each iteration of the greedy parsing procedure, given the input sequence of constituents, (1) node representations are computed for each model by composing the sub-tree representations corresponding to the given model and using its own compositional network (2) each model computes tag scores using its own tagger network (3) tag scores are averaged (4) a coherent path of tag is predicted using the Viterbi algorithm.

	MODEL	< 40			FULL			TIME
		(R)	(P)	F1	(R)	(P)	F1	
GENERATIVE	MAGERMAN (1995)	84.6	84.9	84.8				
	COLLINS (1999)	88.5	88.7	88.6	88.1	88.3	88.2	1247
	CHARNIAK (2000)	90.1	90.1	90.1	89.6	89.5	89.6	
	PETROV AND KLEIN [2007]	90.7	90.5	90.6	90.2	98.9	90.1	307
GENERATIVE WITH RE-RANKING	HENDERSON (2004)				89.8	90.4	90.1	
	CHARNIAK & JOHNSON (2005)			92.0			91.1	
	MCCLOSKEY ET AL (2006)						92.1	
	SOCHER ET AL (2013)			91.1			90.4	390
DISCRIMINATIVE	CARRERAS ET AL. (2008)				90.7	91.4	91.1	
	CHAPTER 5 (V_{10})	90.0	90.1	90.1	89.6	89.7	89.6	
	CHAPTER 5 + DROPOUT (V_{10})	90.6	90.1	90.4	90.2	89.7	89.9	
	THIS WORK	88.8	89.1	89.0	88.2	88.6	88.4	
	THIS WORK + DROPOUT	89.7	90.3	90	89.1	89.9	89.5	30
	THIS WORK + DROPOUT (V_4)	90.5	90.8	90.7	90.1	90.4	90.3	120

Table 6.1: Performance comparison of different state-of-the-art parsers, in terms of Precision (P), Recall (R), and F1 score, for sentences of size ≤ 40 words, and on the full WSJ test set. V_x denotes a voting procedure with x models. The reported time (in seconds) is the time to parse the full WSJ test corpus.

Finally, we report a brief quantitative evaluation of our compositional representations in Table 6.3. Random phrases were picked in the WSJ corpus, and closest neighbors (according to the Euclidean distance) with other phrases of the corpus are reported.

	PP	CLAUSE	DIFF	MOD	NP	CO-ORD	1-WORD		NP	
	ATTACH	ATTACH	LABEL	ATTACH	ATTACH		SPAN	UNARY	INT.	OTHER
MCCLOSKEY ET AL (2006)	0.60	0.38	0.31	0.25	0.25	0.23	0.20	0.14	0.14	0.50
SOCHER ET AL (2013)	0.79	0.43	0.29	0.27	0.31	0.32	0.31	0.22	0.19	0.41
CHAPTER 5	0.74	0.45	0.27	0.25	0.34	0.38	0.24	0.22	0.20	0.57
THIS WORK + DROPOUT	0.78	0.44	0.29	0.27	0.36	0.42	0.24	0.21	0.20	0.60
THIS WORK + DROPOUT (V_4)	0.71	0.43	0.25	0.24	0.35	0.38	0.23	0.21	0.19	0.56

Table 6.2: Detailed parser comparison. We report the average number of bracket errors per sentence for different error categories.

Chapter 6. RNN-Based Phrase Composition for Syntactic Parsing

brendan barba , chairman of the moonachie , n.j. , maker of plastic film products

edmund edelman , chairman of the los angeles county board of supervisors
esther dyson , editor of release 0.0 , an industry newsletter that spots new developments
michael slater , editor of the microprocessor report , an industry newsletter
bruce miller , president of art funding corp. , an art lender
jeffrey nichols , president of apms canada , toronto precious metals advisers ,

eli lilly & co. , indianapolis ,

john kinnard & co. , minneapolis ,
procter & gamble co. , cincinnati ,
anb investment management co. , chicago ,
scimed life systems inc. , minneapolis ,
rjr nabisco inc. 's french cracker subsidiary , belin ,

mr. engelken 's sister , martha , who was born two days before the home run ,

the company 's president , n.j . nicholas , who will eventually be co-chief executive of time warner alongside mr. ross ,
claudio 's sister , isabella , a novitiate in a convent ,
her daughter , elizabeth , an attorney who is vice chairman ,
his brother , parkhaji , whose head is swathed in a gorgeous crimson turban ,
mrs. coleman 's husband , joseph , a physician ,

chairman and chief executive officer

president and chief executive officer
president and chief operating officer
chairman and chief executive
executive vice president and chief financial officer
executive vice president and chief operating officer

Table 6.3: Nearest neighbors (in terms of vector representation Euclidean distance) for several phrases in the WSJ corpus. For every node in the corpus, the sub-tree representations were computed. Then, for the selected phrases, we computed all Euclidean distances. The first phrase is the reference and we report below the 5 top closest phrases in WSJ.

6.4 Conclusion

In this chapter, we introduced a novel RNN-based compositional representation of parsing sub-trees, encoding both the syntactic (tags) and semantic (words) information. The parsing procedure is tightly integrated with the composition operation, and allows us to reach performance of very well-known parsers while (1) adopting a greedy and fast procedure, and (2) avoid standard refined features such as headwords.

7 Syntactic Parsing of Morphologically Rich Language

In chapter 5, we approach syntactic parsing as a succession of phrase prediction problems that are solved recursively in a greedy manner. In chapter 6, this model was enhanced using a novel compositional feature that performed a syntactic and semantic summary of the contents of sub-trees. In this chapter, we introduce a similar composition procedure for the purpose of including morphological information in the context of morphologically rich languages (MRL).

The chapter is organized as follows: we first review the existing methods for parsing MRL. We then describe the proposed morphological composition procedure. We finally provide a comparative evaluation of our approach on a standard parsing task for 9 different languages.

7.1 Introduction

Morphologically rich languages (MRL) are languages for which important information concerning the syntactic structure is expressed through word formation, rather than constituent-order patterns. Unlike English, they can have complex word structure as well as flexible word order. A common practice when dealing with such languages is to incorporate morphological information explicitly [Tsarfaty et al., 2013]. However this poses two problems to the classical generative models: (1) they assume input features to be conditionally independent which makes the incorporation of arbitrary features difficult and (2) refining input features leads to a data sparsity issue. In the other hand, neural network-based models using continuous word representations as input have been able to overcome the data sparsity problem inherent in NLP [Huang and Yates, 2009]. Furthermore, neural networks allow to incorporate arbitrary features and learn complex non-linear relations between them.

In this chapter, we propose to enhance this model for syntactic parsing of MRL, by learning morphological embeddings. We take advantage of a recursive composition procedure similar to the one introduced in Chapter 6 to propagate morphological information during the parsing process. We evaluate our approach on the SPMRL (syntactic parsing of MRL) Shared Task 2014 [Seddah et al., 2013] which provides standardized datasets, evaluation metrics and baseline

results for nine different languages. Each of them comes with a set of morphological features allowing to augment words with information such as their grammatical functions, relation with other words in the sentence, prefixes, affixes and lemmas. We show that integrating morphological features allows to increase dramatically the average performance and yields state-of-the-art performance for a majority of languages.

7.2 Related work

Both the baseline (Berkeley parser) and the current state-of-the-art model on the SPMRL Shared Task 2014 [Björkelund et al., 2014] rely on probabilistic context free grammar (PCFG)-based features. The latter uses a product of PCFG with latent annotation based models [Petrov, 2010], with a coarse-to-fine decoding strategy. The output is then discriminatively re-ranked [Charniak and Johnson, 2005] to select the best analysis. In contrast, the parser used in this chapter constructs the parse tree in a greedy manner and relies only on word, POS tags and morphological embeddings.

Several other papers have reported results for the SPMRL Shared Task 2014. Hall et al. [2014] introduced an approach where, instead of propagating contextual information from the leaves of the tree to internal nodes in order to refine the grammar, the structural complexity of the grammar is minimized. This is done by moving as much context as possible onto local surface features. This work was refined in Durrett and Klein [2015], taking advantage of continuous word representations. The system used in this chapter also leverages words embeddings but has two major differences. First, it proceeds step-by-step in a greedy manner where Durrett and Klein [2015] is using structured inference (CKY). Second, it leverages a compositional node feature which propagates information from the leaves to internal nodes, which is exactly what is claimed not to be done in Durrett and Klein [2015].

Fernández-González and Martins [2015] proposed a procedure to turn a dependency tree into a constituency tree. They showed that encoding order information in the dependency tree makes it isomorphic to the constituent tree, allowing any dependency parser to produce constituents. Like the parser we used, their parser do not need to binarize the treebank as most of the others constituency parsers. Unlike this system, we do not use the dependency structure as an intermediate representation and directly perform constituency parsing over raw words.

7.3 Recurrent greedy parsing

In this chapter, we used the model presented in introduced in Chapters 5 and 6. Each iteration of the procedure merges input constituents into new nodes by applying the following steps. Note that the novelties reside in step 1 and 4:

- 1. **Node tagger:** a neural network sliding window (see Figure 7.2) is applied over the

input sequence of constituents (leaves or heads of trees predicted so far) taking into account the syntactic tags (POS and parsing tags) as well as the morphological tags. This procedure outputs for each constituent a score s_i for each BIOES-prefixed parsing tag $t \in \mathcal{T}$ (\mathcal{T} being the parsing tags ensemble).

- **2. Dynamic programming:** a coherent path of BIOES tags is retrieved by decoding over a constrained graph. This insures (for instance) that a $B-A$ can be followed only by a $I-A$ or a $E-A$ (for all parsing tag A).
- **3. Compositional procedure:** new nodes are created, merging input constituents, according to the dynamic programming predictions. A neural network composition module is then used to compute vector representations for the new nodes, according to the representations of the merged constituents, as well as their corresponding tags (POS or parsing).
- **4. Morphological compositional procedure:** for each new node, a morphological representation is computed for each morphological category. The procedure used to perform this operation is described in Section 7.4.

The procedure ends when the top node is produced.

7.4 Parsing Morphologically Rich Languages

7.4.1 Morphological features

Morphological features enable the augmentation of input tokens with information expressed at a word level, such as grammatical function or relation to other words. For parsing MRL, they have proven to be very helpful [Cowan and Collins, 2005]. The SMPRL corpus provides a different set of morphological features associated to the tree terminals (tokens) for every language. These features include morphosyntactic features such as case, number, gender, person and type, as well as specific morphological information such as verbal mood, proper/common noun distinction, lemma, grammatical function. They also include many language-specific features. For more details about the morphological features available, the reader can refer to Seddah et al. [2013].

7.4.2 Morphological Embeddings

The parser introduced in Chapter 5 and 6 relies only on word and tag embeddings. Besides these features, the proposed model takes advantage of additional morphological features. As illustrated in Figure 7.2, each additional feature m is assigned a different lookup table containing morphological feature vectors of size d_m . The output vectors of the different morphological lookup-tables are simply concatenated to form the input of the next neural network layer.

	\mathbf{I}_W	:	Did	you	hear	the	falling	bombs	?
(a)	\mathbf{I}_T	:	VBD	PRP	VB	DT	VBG	NNS	.
	\mathbf{O}	:	O	S-NP	O	B-NP	I-NP	E-NP	O
<hr/>									
	\mathbf{I}_W	:	Did	r_1	hear	r_2	.		
(b)	\mathbf{I}_T	:	VBD	NP	VB	NP	.		
	\mathbf{O}	:	O	O	B-VP	E-VP	.		
<hr/>									
	\mathbf{I}_W	:	Did	r_1	r_3	?			
(c)	\mathbf{I}_T	:	VDB	NP	VP	.			
	\mathbf{O}	:	B-SQ	I-SQ	I-SQ	E-SQ			

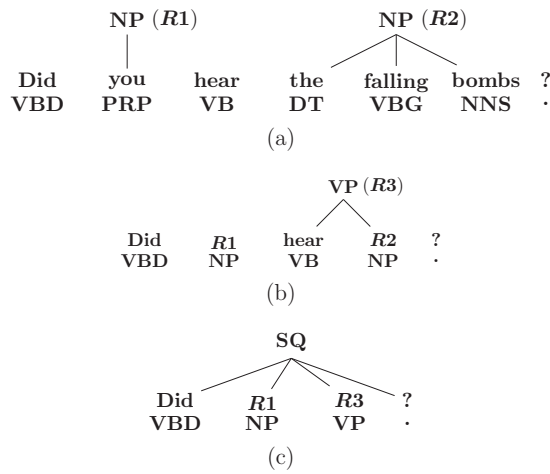


Figure 7.1: Greedy parsing algorithm (3 iterations), on the sentence “Did you hear the falling bombs?”. \mathbf{I}_W , \mathbf{I}_T and \mathbf{O} stand for input words (or composed word representations R_i), input syntactic tags (parsing or part-of-speech) and output tags (parsing), respectively. The tree produced after 3 greedy iterations can be reconstructed as the following: (SQ (VBD Did) (NP (PRP you)) (VP (VB hear) (NP (DT the) (VBG falling) (NNS bombs))) (. ?)).

7.4.3 Morphological composition

Morphological features are available only for leaves. To propagate morphological information to the nodes, we take advantage of a composition procedure similar to the one used in Chapter 6 for words and POS. As illustrated in Figure 7.3, every morphological feature m is assigned a set on composition modules C_{m_i} which take as input i morphological embeddings of dimension d_m . Each composition module perform a matrix-vector operation followed by a non-linearity

$$C_{m_i}(\mathbf{x}) = h(\mathbf{M}_m^i \mathbf{x}) \tag{7.1}$$

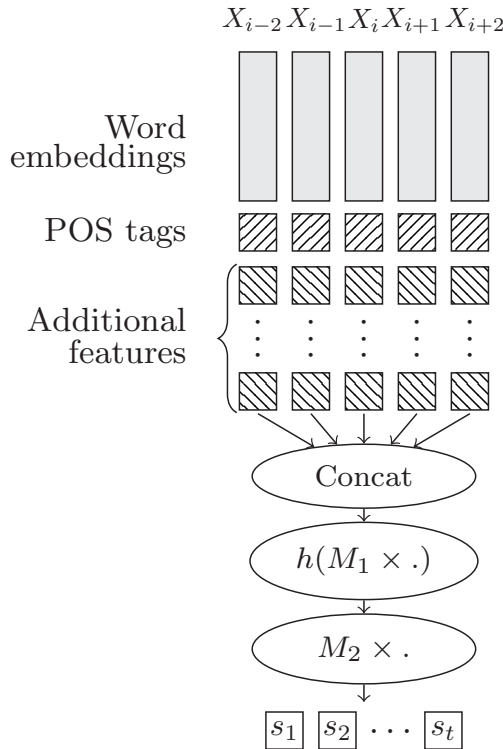


Figure 7.2: A constituent X_i (word or node previously predicted) is tagged by considering a fixed size context window of size K (here $K = 5$). The concatenated output of the compositional history and constituent tags is fed as input to the tagger. A standard two-layers neural network outputs a score s_i for each BIOES-prefixed parsing tag. Additional features can be easily fed to the network. Each category is assigned a new lookup table containing a vector of feature for every possible tag.

where $\mathbf{M}_m^i \in \mathbb{R}^{d_m \times id_m}$ is a matrix of parameters to be trained and $h(\cdot)$ a pointwise non-linearity function. $\mathbf{x} = [\mathbf{x}_1 \dots \mathbf{x}_i]$ is the concatenation of the corresponding input morphological embeddings. Note that given a morphological feature we have a different matrix of weight for every possible size i . In practice most tree nodes do not merge more than a few constituents (see Section 6.2.1).

7.5 Experiments

7.5.1 Corpus

Experiments were conducted on the SPMRL corpus provided for the Shared Task 2014 [Seddah et al., 2013]. It provides sentences and tree annotations for 9 different languages (Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish and Swedish) coming from various sources. For each language, *gold* part-of-speech and morphological tags are provided.

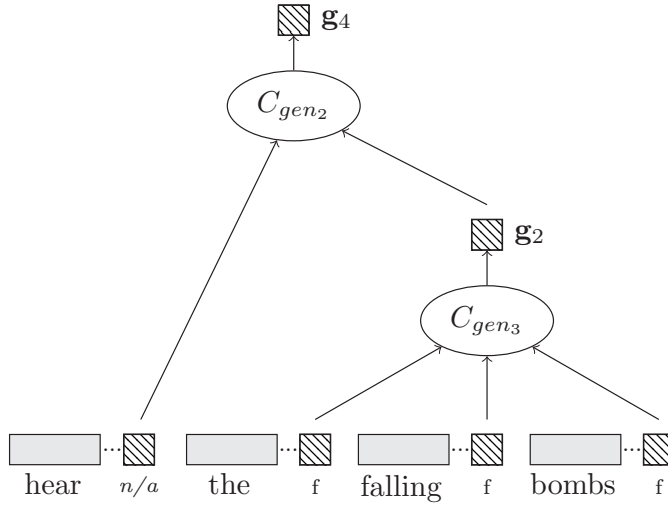


Figure 7.3: Recursive composition of the morphological feature *gender* (male (m) / female (f) / not applicable (n/a)). C_{gen_i} are the corresponding composition modules. The representation \mathbf{g}_2 is first computed using the 3-inputs module C_{gen_3} . \mathbf{g}_4 is obtained by using the 2-inputs module C_{gen_2} .

Results for two baseline baseline system are provided in order to evaluate our models.

7.5.2 Setup

The model was trained using a stochastic gradient descent over the available training data. Hyper-parameters were tuned on the provided validation sets. The word embedding size and POS/parsing tag size were set to $D_{\mathcal{W}} = 100$ and $D_{\mathcal{T}} = 30$, respectively. The morphological tag embedding size was set to 10. The window size of the tagger was set to $K = 7$ and its number of hidden units to 300. All parameters were initialized randomly (including the words embeddings). As suggested in Plaut and Hinton [1987], the learning rate was divided by the size of the input vector of each layer. We applied the same dropout regularization as in Legrand and Collobert [2015].

7.5.3 Results

Table 7.1 presents the influence of adding morphological features to the model. We observe significant improvement for every languages except for Hebrew. On average, morphological features allowed to overcome the original model by 2 F1-score.

Table 7.2 compares the performance in F1-score (obtained with the provided EVALB_SPMRL tool) of different systems, using the provided gold POS and morphological features. We compare our results with the two baselines provided with the task: (1) Berkeley parser with provided POS Tags (Berkeley+POS), (2) Berkeley Parser in raw mode where the parser do its

language	Words + POS	+ morph
Arabic	80.7	82.9
Basque	82.7	90.6
French	81.1	85.0
German	81.5	83.1
Hebrew	91.6	91.5
Hungarian	89.6	90.3
Korean	86.1	86.7
Polish	93.2	93.7
Swedish	81.1	81.5
AVG	85.3	87.3

Table 7.1: Influence of the additional morphological embeddings in terms of F1-score

own POS tagging (Berkeley RAW). We also report the results of the current state-of-the-art model for this task [Björkelund et al., 2014]. We included the same voting procedure as in Chapter 6, using 5 models trained starting from different random initializations. At each iteration of the greedy parsing procedure, the BIOES-tag scores are averaged and the new node representations (words+POS and morphological composition) are computed for each model by composing the sub-tree representations corresponding to the given model, using its own compositional network. One can observe that the proposed model outperforms the best model by 1.1 F1-score on average. Moreover, it yields state-of-the-art performance for 6 among the 9 available languages.

Model	Ara.	Bas.	Fre.	Ger.	Heb.	Hun.	Kor.	Pol.	Swe.	AVG
Berkeley+POS	80.8	76.2	81.8	80.3	92.2	87.6	82.9	88.1	82.9	83.7
Berkeley RAW	79.1	69.8	80.4	79.0	87.3	81.4	73.3	79.5	78.9	78.7
Björkelund et al. [2014]	82.2	90.0	84.0	82.1	91.6	92.6	86.5	88.6	85.1	87.0
Proposed approach	84.1	91.0	85.7	84.6	91.7	91.2	87.8	94.1	82.5	88.1

Table 7.2: Results for all languages in terms of F1-score, using gold POS and morphological tags. Berkeley+POS and Berkeley RAW are the two baseline system results provided by the organizers of the shared task. Our experiments used an ensemble of 5 models, trained starting from different random initializations.

Finally, Table 7.3 compares the performance of different systems for a more realistic parsing scenario where the gold POS and morphological tags are unknown. For these experiments, we use the same tags as in Björkelund et al. [2014]¹, obtained using the freely available tool MarMoT [Mueller et al., 2013]. We compare our results with the same model as for the gold tags experiences. Additionally, we compare our results with two recent models reporting results for the SPMRL Shared Task 2014. We see that the proposed model yields state-of-the-art performance for 4 out of 9 available languages.

¹The tags used are available here: <http://cistern.cis.lmu.de/marmot/models/CURRENT/>

Chapter 7. Syntactic Parsing of Morphologically Rich Language

Model	Ara.	Bas.	Fre.	Ger.	Heb.	Hun.	Kor.	Pol.	Swe.	AVG
Berkeley+POS	78.7	74.7	79.8	78.3	85.4	85.2	78.6	86.7	80.6	80.9
Berkeley RAW	79.2	70.5	80.4	78.3	87.0	81.6	71.4	79.2	79.2	78.5
Durrett and Klein [2015]	80.2	85.4	81.2	80.9	88.6	90.7	82.2	93.0	83.4	85.1
Fernández et al. [2015]	<i>n/a</i>	85.9	78.7	78.7	89.0	88.2	79.3	91.2	82.8	84.2
Björkelund et al. [2014]	81.3	87.9	81.8	81.3	89.5	91.8	84.3	87.5	84.0	85.5
Proposed approach	80.4	87.5	80.8	82.0	91.6	90.0	84.8	93.0	80.5	85.6

Table 7.3: Results for all languages in terms of F1-score using predicted POS and morphological tags. Berkeley+POS and Berkeley RAW are the two baseline system results provided by the organizers of the shared task. Our experiments used an ensemble of 5 models, trained starting from different random initializations.

7.6 Conclusion

In this chapter, we proposed to extend the parser introduced in Chapter 6 by learning morphological embeddings. We take advantage of a recursive procedure to propagate morphological information through the tree during the parsing process. We showed that using the morphological embeddings boosts the F1-score and allows the current state-of-the-art model on the SPMRL Shared Task 2014 corpus to be outperformed. Furthermore, our approach obtains state-of-the-art performance for a majority of languages.

8 Conclusion

The work in this thesis has addressed some of the challenges in natural language processing (NLP). Automatic understanding of natural language is especially difficult due to the many ambiguities that are encountered as well as the complex reasoning and general knowledge about the world that are required. The number of possible sentences is virtually unlimited and the same idea can be expressed in many different ways. Conversely, the same sentence may be interpreted in different ways. While the first attempts to capture the meaning of natural language sentences using computers were based on sets of hand-crafted rules, it is infeasible to cover all possibilities using such rules.

With the advent of statistical modeling, rules came to be automatically learned through the analysis of a large corpora of examples. These techniques offer many advantages compared to hand-crafted rules, e.g. learning commonly and rarely occurring rules which would otherwise require an enormous effort from language experts. However, these approaches mostly rely on hand-crafted input features.

This thesis has approached natural language processing using neural networks that simultaneously learn *how to solve a given NLP task* and *how to extract relevant information* for that task. This is possible using models that learn to predict outputs from inputs in an end-to-end manner. A key element in our work has been to use continuous representations, known as word embeddings, rather than discrete representations of words.

Throughout our work, continuous representations are used to represent words as well as sentence segments for the purpose of solving several NLP tasks. In Chapter 3, we have investigated convolutional neural networks (CNN) to extract locally-relevant representations of words from a fixed-size context window. In Chapter 4, we have introduced a compositional procedure that maps sentence segments of arbitrary size into fixed-size representations. In Chapter 5, we have introduced a novel greedy approach to syntactic parsing casting the task as a succession of phrase classification tasks. Finally, this approach has been enhanced in Chapters 6 and 7 using the compositional procedure introduced in Chapter 4 to represent sub-trees as continuous vectors.

The key contributions of this thesis can be drawn along three axes:

- **End-to-end training:**

We have empirically established that end-to-end training of neural networks is effective at solving various NLP tasks with minimal use of prior knowledge. The proposed neural network models using few basic features as inputs were found to obtain performance comparable to traditional NLP statistical approaches across the several tasks explored in the thesis. As an example, we showed that the deterministic head-word procedure can be replaced by a smooth vector representation that is learned along with the model obtaining better performance. The end-to-end approach was validated on several NLP tasks such as multiword expression tagging, constituency parsing obtaining state-of-the-art results in all of them.

- **Sequence segment representations:**

We have shown that word embeddings can be combined in order to produce effective continuous representations of sequence segments. This approach has been explored for two types of neural networks:

- *CNN-based representations:* A CNN has been used to extract fixed-sized continuous representations that locally focus on different regions of a sentence. A standard generative model, namely FastAlign, has been outperformed by this approach on a bilingual word alignment task.
- *RNN-based representations:* A novel compositional operation enables the representation of arbitrarily-sized sentence segments as fixed-size vectors. By applying such operation recursively, parsing trees can be represented as fixed-size vectors as well. We showed that such representations are able to produce state-of-the-art performance for a syntactic parsing task. These vectors are likely to summarize syntactic and semantic information.

In the context of morphologically rich languages, a similar compositional operation has been successfully applied to propagate morphological information through the parsing process. This approach has obtained state-of-the-art performance for a majority of the 9 languages considered.

- **Task-oriented evaluation:**

In this thesis, we approached several NLP tasks using the proposed deep neural network models. The conclusion for each of these tasks is detailed below:

- **Bilingual word alignment** for machine translation.

We have presented a simple neural network alignment model trained on unlabeled data based on an *aggregation operation* borrowed from the computer vision literature. The proposed architecture extracts representations of windows around target and source words using CNN, obtaining alignment scores using simple dot

products between representations. This approach has outperformed the popular FastAlign model significantly. On a full machine translation task, significant improvements in terms of BLUE score have been obtained.

- **Multiword expression:** We have introduced a novel approach to the phrase prediction task. The proposed approach learns fixed-size continuous representations of arbitrarily-sized chunks by composing word embeddings. These are then used to directly identify and classify phrases. We showed that our approach outperforms the best performing model for this task published to this date.
- **Syntactic parsing:** We have proposed a greedy approach that casts syntactic parsing as a succession of phrase prediction problems. This approach uses a RNN-based compositional representation of parsing sub-trees. We have shown that our approach achieves performance comparable to that of popular parsers while avoiding standard refined features, e.g. headwords, on a task of syntactic parsing of English sentences. A full implementation of our approach has been made available online¹. In the context of morphologically rich languages, a similar procedure that learns morphological embeddings has been shown to obtain state-of-the-art performance for a majority of languages.

In summary, this thesis has shown that neural network architectures can be successfully used for a wide range of NLP tasks and applications without using expert linguistic knowledge. Furthermore, architectures have been shown to be mostly reusable across tasks. We believe that the representation power of continuous vectors has contributed to this goal to a large extent.

8.1 Future research

Word alignment model:

In statistical machine translation, the state-of-the-art system in machine translation use phrase-based techniques [Koehn et al., 2003]. A research direction would explore phrase alignments exploiting both aggregation and continuous phrase representations. This would enormously simplify the computation of phrase alignments. Better phrase alignments, leading to better phrase translations, would be possible as well. The composition procedure introduced in Chapter 4 and further exploited in Chapters 6 and 7, along with an alignment model similar to the one introduced in Chapter 3 could be readily used.

Syntactic Parsing:

The proposed parsing system could be improved along the following directions:

- Improving sub-tree-representations: Auto-encoder techniques [Vincent et al., 2008]

¹<http://joel-legrand.fr>

Chapter 8. Conclusion

might be explored to improve the quality of sub-tree vector representations. In our case, sub-tree representations could be trained to recover the words spanned. This idea could be further developed by asking the autoencoder to recover both sub-tree structure and parsing tags.

- Exploring self-training techniques: State-of-the-art parsing systems such as McClosky et al. [2006] exploit self-training techniques in which a parser is used to tag unlabeled dataset first to be used as ground truth later. Such techniques allow a large amount of unlabeled data to be exploited.
- Exploring “less greedy” methods: One of the strengths of our parsing system resides in its greedy nature. Nonetheless, less greedy methods considering multiple decoding paths during training would be worth exploring. For this purpose, global scores for trees would be required in order to discriminate between different solutions, in the spirit of Socher et al. [2013a] as performed for a parse re-ranking task.

Bibliography

- M. A. Alonso, J. Vilares, and V. M. Darriba. On the usefulness of extracting syntactic dependencies for text indexing. In *Artificial Intelligence and Cognitive Science*, pages 3–11. 2002.
- M. Baroni and R. Zamparelli. Nouns are vectors, adjectives are matrices representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP), 2010.
- R. Battiti. First- and second-order methods for learning: Between steepest descent and newton’s method. *Neural Computation*, 4(2):141–166, 1992.
- Y. Bengio, R. Ducharme, and P. Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.
- Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
- Z. Bitvai and T. Cohn. Non-linear text regression with a deep convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 180–185, 2015.
- A. Björkelund, O. Cetinoglu, A. Falenska, R. Farkas, T. Mueller, W. Seeker, and Z. Szántó. Introducing the ims-wrocław-szeged-cis entry at the SPMRL 2014 shared task: Reranking and morpho-syntax meet unlabeled data. *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, 2014.
- W. Blacoe and M. Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.
- P. Blunsom and T. Cohn. Discriminative word alignment with conditional random fields. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*, 2006.

Bibliography

- O. Bojar and M. Prokopová. Czech-English Word Alignment. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, 2006.
- A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014.
- L. Bottou. *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. PhD thesis, 1991.
- L. Bottou, Y. LeCun, and Y. Bengio. Global training of document processing systems using graph transformer networks. In *Proceedings of Computer Vision and Pattern Recognition*, 1997.
- O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*. 2008.
- J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Advances in Neural Information Processing Systems (NIPS)*. 1990.
- P. F. Brown, J. Cocke, S. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A Statistical Approach to Machine Translation. *Computational Linguistics*, 1990.
- P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 1992.
- X. Carreras, M. Collins, and T. Koo. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CNLL)*, pages 9–16, 2008.
- E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL)*, pages 132–139, 2000.
- E. Charniak and M. Johnson. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180, 2005.
- D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 740–750, 2014.
- Y. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*, 2013.

- Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, volume 1, pages 167–176, 2015.
- Z. Chenxi, Q. Xipeng, C. Xinchu, and H. Xuanjing. A re-ranking model for dependency parser with recursive convolutional neural network. *arXiv preprint arXiv:1505.05667*, 2015.
- D. Chiang. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 2007.
- K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014a.
- K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. 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)*, 2014b.
- N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 1956.
- J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- S. Clark, B. Coecke, and M. Sadrzadeh. A compositional distributional model of meaning. *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pages 133–140, 2008.
- M. Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1–8, 2002.
- M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 2003.
- R. Collobert. Deep learning for efficient discriminative parsing. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, pages 224–232, 2011.
- R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 160–167, 2008.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2011.

Bibliography

- M. Constant, M. Candito, and D. Seddah. The LIGM-Alpage Architecture for the SPMRL 2013 Shared Task: Multiword Expression Analysis and Dependency Parsing. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 46–52, 2013.
- F. Costa, P. Frasconi, V. Lombardo, and G. Soda. Towards incremental parsing of natural language using recursive neural networks. *Applied Intelligence*, 19(1-2):9–25, 2002.
- B. Cowan and M. Collins. Morphology and reranking for the statistical parsing of spanish. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 795–802, 2005.
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCS)*, 1989.
- P. S. Dhillon, D. Foster, and L. Ungar. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems (NIPS)*, pages 199–207, 2011.
- L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, 2014.
- L. Dong, F. Wei, M. Zhou, and K. Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.
- C. N. dos Santos and B. Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1818–1826, 2014.
- C. N. dos Santos, B. Xiang, and B. Zhou. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*, 2015.
- G. Durrett and D. Klein. Neural CRF parsing. In *Proceedings of the Association for Computational Linguistics*, 2015.
- C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 644–648, 2013.
- J. Earley. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery (CACM)*, 13(2):94–102, 1970.
- J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

- J. L. Elman. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225, 1991.
- D. Fernández-González and A. F. T. Martins. Parsing as reduction. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.
- J. R. Finkel, A. Kleman, and C. D. Manning. Efficient, feature-based, conditional random field parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, volume 46, pages 959–967, 2008.
- L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppín. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.
- J.R. Firth. A synopsis of linguistic theory 1930-1955. *Studies in linguistic analysis*, 1957.
- J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2–13. Association for Computational Linguistics, 2014.
- J. Giménez and L. Màrquez. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2004.
- C. Goller and A. Küchler. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 347–352, 1996.
- E. Grefenstette, G. Dinu, Y. Zhang, M. Sadrzadeh, and M. Baroni. Multi-step regression learning for compositional distributional semantics. *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, 2013.
- D. Hall, G. Durrett, and D. Klein. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.
- Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.
- K. Hashimoto, M. Miwa, Y. Tsuruoka, and T. Chikayama. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1372–1376, 2013.
- J. Henderson. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, 2003.
- J. Henderson. Discriminative training of a neural network statistical parser. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, 2004.

Bibliography

- K. M. Hermann and P. Blunsom. The role of syntax in vector space models of compositional semantics. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 894–904, 2013.
- F. Hill, K. Cho, S. Jean, C. Devin, and Y. Bengio. Not all neural embeddings are born equal. *arXiv preprint arXiv:1410.0718*, 2014.
- G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Distributed Representations. 1986.
- G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.
- S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidh. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *Field Guide to Dynamical Recurrent Networks*. 2001.
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989.
- F. Huang and A. Yates. Distributional representations for handling sparsity in supervised sequence-labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, 2009.
- Z. Huang, M. Harper, and S. Petrov. Self-training with products of latent variable grammars. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2010.
- O. Irsoy and C. Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2014.
- M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 633–644, 2014a.
- M. Iyyer, P. Enns, J. L. Boyd-Graber, and P. Resnik. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1113–1122, 2014b.
- R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

-
- R. Józefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.
- N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, 2014.
- T. Kasami. An efficient recognition and syntax analysis algorithm for context-free languages. Technical report, 1965.
- M. Kay. Readings in natural language processing. chapter Algorithm Schemata and Data Structures in Syntactic Processing. 1986.
- Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- D. Klein and C. D. Manning. Conditional structure versus conditional estimation in nlp models. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002.
- P. Koehn, F. J. Och, and D. Marcu. Statistical Phrase-based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (NAACL)*, pages 48–54, 2003.
- P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Proceedings of the Association for Computational Linguistics (ACL), pages 177–180, 2007.
- T. Kudoh and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, 2000.
- J. K. Kummerfeld, D. Hall, J. R. Curran, and D. Klein. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Eighteenth International Conference on Machine Learning, ICML*, 2001.
- T. K. Landauer and S. T. Dumais. A solution to plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 1997.

Bibliography

- P. Le and W. Zuidema. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.
- Y. Le Cun. A learning scheme for asymmetric threshold networks. In *Proceedings of Cognitive* 85, 1985.
- R. Lebrecht and R. Collobert. Word embeddings through hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014.
- Y. Lecun. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989.
- J. Legrand and R. Collobert. Recurrent greedy parsing with neural networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2014.
- J. Legrand and R. Collobert. Joint RNN-based greedy parsing and word composition. In *International Conference on Learning Representations (ICLR)*, 2015.
- J. Legrand and R. Collobert. Deep neural networks for syntactic parsing of morphologically rich languages. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2016a.
- J. Legrand and R. Collobert. Phrase representations for multiword expressions. In *ACL 2016 12th Workshop on Multiword Expressions (MWE 2016)*, 2016b.
- J. Legrand, M. Auli, and R. Collobert. Neural network-based word alignment through score aggregation. In *ACL 2016 First Conference on Machine Translation (WMT16)*, 2016.
- O. Levy and Y. Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 302–308, 2014.
- O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, pages 211–225, 2015.
- J. Li, R. Li, and E. H. Hovy. Recursive deep models for discourse parsing. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 2061–2069, 2014.
- Y. Liu, F. Wei, S. Li, H. Ji, M. Zhou, and H. Wang. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290, 2015.

- D. M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 1995.
- J. Martin, R. Mihalcea, and T. Pedersen. Word Alignment For Languages With Scarce Resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts (WPT)*, 2005.
- D. McClosky, E. Charniak, and M. Johnson. Effective self-training for parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006.
- W. S. McCulloch and W. Pitts. Neurocomputing: Foundations of research. chapter A Logical Calculus of the Ideas Immanent in Nervous Activity. 1988.
- D. I. Melamed. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Third Workshop on Very Large Corpora*, 1995.
- R. Mihalcea and T. Pedersen. An Evaluation Exercise for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond - Volume 3*, pages 1–10, 2003.
- T. Mikolov. *Statistical language models based on neural networks*. PhD thesis, PhD thesis, Brno University of Technology. 2012, 2012.
- T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *Proceedings of Interspeech*, volume 2, page 3, 2010.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.
- T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*. 2013b.
- J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive Science*, 2010.
- A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21*. 2009.
- R. C. Moore. A Discriminative Framework for Bilingual Word Alignment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88, 2005.
- T. Mueller, H. Schmid, and H. Schütze. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Bibliography

- T. H. Nguyen and R. Grishman. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 365–371, 2015.
- A. B. J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1963.
- F. J. Och and H. Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 2003.
- D. Palaz, G. Synnaeve, and R. Collobert. Jointly Learning to Locate and Classify Words using Convolutional Networks. In *Proceedings of Interspeech*, page to appear, 2016.
- J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL <http://www.aclweb.org/anthology/D14-1162>.
- S. Petrov. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, 2010.
- S. Petrov and D. Klein. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, 2007.
- S. Petrov and D. Klein. Sparse multi-scale grammars for discriminative latent variable parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.
- P. O. Pinheiro and R. Collobert. From Image-level to Pixel-level Labeling with Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1713–1721, 2015.
- D. C. Plaut and G. E. Hinton. Learning sets of filters using back-propagation. *Computer Speech and Language*, 1987.
- J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 1990.
- V. Punyakanok, D. Roth, and W. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 2008.
- L. R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. 1990.
- F. Radu, I. Abe, J. Hongyan, and Z. Tong. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, pages 168–171, 2003.

- A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, February 1999.
- F. Rosenblatt. The perceptron: a perceiving and recognizing automaton. Technical report, 1957.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*. 1986.
- E. F. T. K. Sang and J. Veenstra. Representing text chunks. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, 1999.
- T. Schnabel, I. Labutov, D. Mimno, and T. Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2015.
- D. Seddah, R. Tsarfaty, S. Kübler, M. Candito, J. D. Choi, R. Farkas, J. Foster, I. Goenaga, K. Gójenola, Y. Goldberg, S. Green, N. Habash, M. Kuhlmann, W. Maier, J. Nivre, A. Przepiórkowski, R. Roth, W. Seeker, Y. Versley, V. Vincze, M. Woliński, A. Wróblewska, and E. Villemonte De La Clergerie. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, 2013.
- R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2011a.
- R. Socher, C.C.Y Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 129–136, 2011b.
- R. Socher, J. Bauer, C. Manning, and A. Ng. Parsing With Compositional Vector Grammars. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 455–465. 2013a.
- R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013b.
- X. Sun, L. Morency, D. Okanohara, and J. Tsujii. Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, 2008.
- M. Sundermeyer, R. Schlüter, and H. Ney. LSTM neural networks for language modeling. In *Proceedings of Interspeech*, 2012.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*, 2014.

Bibliography

- A. Tamura, T. Watanabe, and E. Sumita. Recurrent Neural Networks for Word Alignment Model. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 1470–1480, 2014.
- B. Taskar, S. Lacoste-Julien, and D. Klein. A Discriminative Matching Approach to Word Alignment. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2005.
- K. Toutanova, T. H. Ilhan, and C. D. Manning. Extensions to hmm-based statistical word alignment models. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, 2002.
- R. Tsarfaty, D. Seddah, S. Kübler, and J. Nivre. Parsing morphologically rich languages: Introduction to the special issue. *Computational Linguistics*, 2013.
- Y. Tsvetkov, M. Faruqui, W. Ling, G. Lample, and C. Dyer. Evaluation of word vector representations by subspace alignment. pages 2049–2054, 2015.
- J. Turian and I. Dan Melamed. Advances in discriminative parsing. In *Proceedings of the Joint International Conference on Computational Linguistics and Association of Computational Linguistics (COLING/ACL)*, 2006.
- J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.
- K. Uchimoto, Q. Ma, M. Murata, H. Ozaku, and H. Isahara. Named entity extraction based on a maximum entropy model and transformation rules. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 2000.
- J. Väyrynen and T. Honkela. Word category maps based on emergent features created by ICA. In *Proceedings of the STeP'2004 Cognition + Cybernetics Symposium*, 2004.
- P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, Proceedings of the International Conference on Machine Learning (ICML), pages 1096–1103, 2008.
- O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.
- S. Vogel, H. Ney, and C. Tillmann. HMM-Based Word Alignment in Statistical Translation. In *Proc. of COLING*, 1996.
- P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 352–357, 2015.

- L. Wittgenstein. *Philosophical Investigations*. (Translated by Anscombe, G.E.M.). 1953.
- W. Xu, M. Auli, and S. Clark. Ccg supertagging with a recurrent neural network. *Proceedings of the Association for Computational Linguistics (ACL)*, 2:250–255, 2015.
- N. Yang, S. Liu, M. Li, M. Zhou, and N. Yu. Word Alignment Modeling with Context Dependent Deep Neural Network. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 166–175, 2013.
- W. Yin and H. Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 901–911, 2015.
- D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344, 2014.
- M. Zhu, Y. Zhang, W. Chen, M. Zhang, and J. Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 434–443, 2013.
- A. Zollmann and A. Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, 2006.

Legrand Joël

Curriculum Vitae

Education

- 2012-present **Ph.D. in Electrical Engineering**, *École Polytechnique Fédérale de Lausanne*, Lausanne, Switzerland.
Under the supervision of Ronan Collobert
- 2011-2012 **M.Sc. in Fundamental Cognitive Science**, *Université Lyon 2*, Lyon, France.
- 2009-2011 **M.Sc. degree in Computer Science**, *Université de Lorraine*, Nancy, France.
Research Master “Recognition, Learning and Reasoning”
Master Thesis, *Developmental Reinforcement Learning for Robotics*.
Under the supervision of Alain Dutech
- 2005-2009 **Bachelor’s degree in Mathematics and Computer Science**, *Université de Lorraine*, Nancy, France.

Experience

- 2015 **Research Intern at Facebook**, *Facebook AI Research*, Menlo Park (CA), USA,
Under the supervision of Ronan Collobert.
During my PhD, I did a three months internship at Facebook AI Research. My work focused on Machine Translation and more precisely on Bilingual Word Alignment using Convolutional Neural Networks.
- 2012–Present **Research Assistant**, *Idiap Research Institute*, Martigny, Switzerland,
Under the supervision of Ronan Collobert.
I am currently doing a PhD at the Idiap Research Institute. My research focuses on Natural Language Processing using Deep Neural Networks and more precisely on Syntactic Parsing and Sentence Representations.
- 2010–2011 **Research Intern**, *Laboratoire Lorrain de Recherche en Informatique et ses applications (LORIA)*, Nancy, France,
Under the supervision of Alain Dutech.
As a member of the team MAIA (Autonomous Intelligent MACHines), I had to explore a developmental approach to learning a robotics task where the perception and motor skills of the robot were growing in richness and complexity during learning. This was done using the Reinforcement Learning framework and tested using Khepera III robots for a simple orientation task in a maze.

Les Marais, 10 – 1922 Salvan, Switzerland

☎ (0041) 76 703 99 77 • ✉ joel.legrand@idiap.ch

📄 people.idiap.ch/jlegrand

2009–2010 **Research Intern**, *Laboratoire Lorrain de Recherche en Informatique et ses applications (LORIA)*, Nancy, France,
Under the supervision of Brigitte Wrobel-Dautcourt.
During this internship, I laid the foundation stone of artEoz, a didactic software intended for the first-year students in computer science. This software allows to visualize the memory when running a program written in Java or Python. It is freely available at this address: <http://artez.loria.fr/>.

Computer skills

Languages

Intermediate Python, Perl, java

Advanced Lua, C, Shell, L^AT_EX

Scientific tools

Intermediate Matlab, TensorFlow

Advanced Torch, Moses

Publications

2016 J. Legrand, M. Auli, R. Collobert, **Neural Network-based Word Alignment through Score Aggregation**, *In ACL 1st Conference on Machine Translation (WMT 2016)*

J. Legrand, R. Collobert, **Phrase Representations for Multiword Expressions**, *In ACL Workshop on Multiword Expressions (MWE 2016)*

J. Legrand, R. Collobert, **Deep Neural Networks for Syntactic Parsing of Morphologically Rich Languages**, *In ACL 2016*

2015 J. Legrand, R. Collobert, **Joint RNN-Based Greedy Parsing and Word Composition**, *In ICLR, 2015*

2014 J. Legrand, R. Collobert **Recurrent Greedy Parsing with Neural Networks**, *In ECML, 2014*

2013 R. Lebrete, J. Legrand, R. Collobert **Is Deep Learning Really Necessary for Word Embeddings?**, *In NIPS Workshop on Deep Learning, 2013*

Languages

French **Mother tongue**

English **Fluent**

Spanish **Basic**

Interests

- Music (Guitar, Bouzouki, Accordion, Irish flute)

- Free Software

