TÉCNICO LISBOA

# Understanding Spatial Semantics in Natural Language

**Alexey Mazalov**

Thesis to obtain the Master of Science Degree in

## Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. David Manuel Martins de Matos

### Examination Committee

Chairperson: Prof. Miguel Nuno Dias Alves Pupo Correia
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. Paulo Miguel Torres Duarte Quaresma

**October 2016**

# Acknowledgments

I would like to thank my MSc thesis supervisors, Prof. Bruno Martins and Prof. David Matos, for always being ready to share their knowledge with me, for reading the preliminary versions of my thesis and for respecting my style of work. They were always ready to adjust to it, giving me many useful tips on how to achieve better results and make my investigations more productive. Even on weekends and public holidays, they were available for advice.

I also give many thanks to the MEIC program coordinator Prof. Miguel Pupo Correia and his assistant Dr. Lurdes Farrusco for their guidance over the two years of my studies at IST and for understanding of the difficulties, that an international student faces while staying abroad.

I am full of gratitude to Prof. Victor Timofeev and Prof. Maria Vinogradova from the Bauman Moscow State Technical University (BMSTU), who encouraged me to accept the challenge of abandoning the boundaries of the previously established comfort zone and to face the challenge of studying in Portugal. I give many thanks to Sofia Malikova, member of the International Cooperation and Mobility Office of BMSTU, for helping me to settle the formalities while getting prepared for leaving Moscow for Lisbon. I thank the members of the International Cooperation and Mobility Office of IST for having stayed in contact with me since the earliest days of submitting my candidacy for participating in the TIME international mobility program until the present day.

I thank all my University Professors, both in Portugal and in Russia, for helping me to acquire all the professional skills that I currently possess.

Finally, I can't thank enough my parents, who always gave me full moral and largely material support while I was pursuing my studies.

To each and every one of you – Thank you.

# Abstract

In this MSc thesis we focus on the Spatial Role Labeling (SpRL) task, which emerged as a variation of Semantic Role Labeling (SRL). The approach we advocate for SpRL fully exploits its kinship with SRL. It is based on adopting the NLPNET system, that is built on a multi-layer convolutional neural network and has proved its efficiency in dealing with the SRL task. We demonstrate that SpRL can be performed by a system originally designed for SRL, as well as that by applying several relatively simple input format transformations, it is possible to achieve a gain in the system's performance. We present the results for the distinct cases of static and dynamic spatial relations. Although being relatively modest, the performance demonstrated by the system is comparable with that of the known state-of-the-art systems.

# Keywords

Machine Learning, Convolutional Neural Network, Natural Language Semantics, Spatial Role Labeling, NLPNET.

# Resumo

A presente tese de mestrado insere-se na tarefa de Anotação de Papeis Espaciais (Spatial Role Labeling, SpRL) que surgiu como um caso particular da tarefa de Anotação de Papeis Semânticas (Semantic Role Labeling, SRL). A abordagem que preconizamos benefícia ao máximo da afinidade entre as duas tarefas mencionadas. A fim de lidar com a SpRL, procura-se reconfigurar o sistema NLPNET que tem como parte fundamental uma rede neuronal convolucional. O dito sistema já provou ser eficaz no âmbito da tarefa de SRL. Demonstramos que a anotação de papeis espaciais pode ser abordada por um sistema inicialmente desenhado para a SRL. Também visamos exibir que com a aplicação de várias transformações relativamente simples ao formato de dados de entrada, usadas na fase de treino de modelos, é possível melhorar a qualidade dos mesmos. Os resultados para dois casos distintos, nomeadamente para relações espacias estáticas e dinâmicas, são apresentados. Embora seja relativamente moderado, o desempenho do sistema é comparável com o dos sistemas de ponta conhecidos.

# Palavras Chave

Aprendizagem Automática, Redes Neuronais Convolucionais, Semântica em Linguagem Natural, Anotação de Papeis Espaciais, NLPNET.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **CoNLL** | Conference on Natural Language Learning |
| **CRF** | Conditional Random Fields |
| **HAL** | Hyperspace Analogue to Language |
| **HMM** | Hidden Markov Model |
| **MLP** | Multilayer Perceptron |
| **NLP** | Natural Language Processing |
| **POS** | Part-Of-Speech |
| **RAM** | Random Access Memory |
| **SemEval** | Semantic Evaluation |
| **SpRL** | Spatial Role Labeling |
| **SRL** | Semantic Role Labeling |
| **SVM** | Support Vector Machines |
| **VIEW** | Visually Informed Embeddings of Words |

# 1

# Introduction

## Contents

## 1.1 Motivation and Context

Spatial Role Labeling (SpRL) is one of the Natural Language Processing (NLP) tasks, that center on attributing meaning to a piece of text.

The thrive in computational capacity in the late 1980s led to an extensive recognition of machine learning algorithms, rather than hard-coded rules, as a means of tackling most of the NLP tasks. The fact that these algorithms require considerable computational power stopped being a deterrent for their use. The primary spheres of interest of the research community gradually shifted from simple word sense disambiguation to more advanced semantic analysis tasks, such as Semantic Role Labeling (SRL), Temporal Annotation, Sentiment Analysis and SpRL among others.

The most significant events, aimed at discussing and comparative evaluation of the advances in the spheres of NLP, are the Senseval congresses, later succeeded by the Semantic Evaluation (SemEval) series of workshops, and the Conference on Natural Language Learning (CoNLL) shared tasks. Since the first Senseval congress in 1997, the latter have been focused primarily on semantic analysis tasks, such as SRL, while the former have touched a wide range of NLP problems, but have recently been dedicated to Semantic Role Labeling, Temporal Annotation and Spatial Role Labeling.

The Semantic Role Labeling task deserves a special mention because it is related to capturing sentence meaning, making use of approaches that are similar to the ones we intend to employ when addressing Spatial Role Labeling. SRL was first in focus at the Senseval-3 congress [2] and it was also the main topic of a series of CoNLL tasks [3–5]. The aim of the task is to detect, predict and attribute a semantic role to each word or phrase in a sentence. The possible roles include: *agent* (i.e., the doer of action), *predicate* (usually a verb designating the action), *beneficiary*, etc. A number of solutions were conceived to deal with SRL. Alongside systems, relying on traditional approaches, based on lexical and syntactic features, there is a class of systems, leveraging modern machine learning techniques, such as a unified multilayer neural network architecture [1]. These novel ideas were implemented in SENNA, which is a system capable of solving the tasks of semantic role labeling, chunking, name entity recognition and Part-Of-Speech (POS) tagging.

Spatial Role Labeling can be defined as the task of mapping spatial relations between objects, provided in the form of text, onto a formal representation. The basic static spatial roles, assigned to the words/phrases in a sentence are the *spatial indicator*, the *trajector* and the *landmark*. For a more general case of dynamic spatial relations it is necessary to consider the *motion indicator*, the *path*, the *distance* and the *direction* as the possible spatial roles. Some of the first achievements in SpRL were presented by Parisa Kordjamshidi et al. [6] with a system based on the application of machine-learning techniques.

SpRL is actually a variation of SRL, but has barely been treated as such. On the contrary, the known existing SpRL systems strongly rely on explicit linguistic knowledge, heuristics and the output of systems for other NLP tasks used as auxiliary data.

## 1.2 Thesis Proposal

Taking inspiration on the existing SpRL systems, as well as on the previous approaches used for SRL, we intend to adapt existing solutions for the SRL task, which leverage modern machine learning techniques such as *convolutional neural networks*, for performing SpRL.

The performed analysis over systems, addressing the SRL and SpRL tasks, reveals a great degree of similarity between these tasks. Both of them are concerned with attributing meaning to pieces of text in a natural language. Analogously to SRL, the SpRL task implies automatic identification of semantic relations and expressing them using an accepted formalism, reflecting the task's nature and the underlying assumptions.

The cornerstone of SpRL consists in identifying the spatial indicator, serving as a pivot for the spatial relation, which is conceptually very similar to determining the predicate, playing the semantic pivot role for SRL. It is perceived that a semantic parser, capable of successful tackling the basic version of the SpRL task, should output a set of triplets (i.e., $trajector$, $spatial\_indicator$ and $landmark$), one for each relation expressed in a sentence given in unrestricted natural language. Similarly, a good SRL parser should produce a set of tuples, reflecting the predicate and its most important arguments, captured from a sentence in unrestricted natural language.

The overview of the previously created systems for SpRL has drawn us to the conclusion that there is still much to be done in the field. The systems that participated in the recent SemEval challenges showed moderate results in terms of performance indicators and domain independence. The main hindering factor was probably the limited size of the training corpus [7]. The KUL system relied on a set of linguistically-motivated features for spatial pivot identification, and it used the results of SRL and dependency parsing as auxiliary data for arguments classification. At the SemEval-2012 shared task, the UTDSpRL [8] system managed to outperform KUL slightly, at the cost of leveraging high recall heuristics for selecting candidate tuples.

We believe that the kinship of the SpRL task with SRL was not previously fully exploited. By adopting past achievements in the SRL task for the needs of SpRL, we aim at making use of the state-of-the-art approaches and tools, that have already proven their effectiveness (the F1 score, achieved by state-of-the-art SRL parsers, has risen as high as 0.8). Moreover, we believe, that our work is a step away from feature engineering using explicit linguistic knowledge for SpRL, which falls in line with the tendencies that prevail recently in other NLP tasks.

We found the multilayer convolutional neural network architecture, used by famous SRL systems, such as SENNA [1] and NLPNET [9], promising for addressing the SpRL task. The latter system was deemed especially suitable for being reconfigured to deal with SpRL, due to its improved performance characteristics and higher specialization on the SRL task. Despite that, the system is flexible in terms of labeling input phrases with any tags, i.e., there is no need that the tags be taken from a pre-defined

set of generally-recognized semantic role labels. By leveraging the three-step approach (i.e., separate networks for spatial pivot identification, spatial argument identification and argument classification), we intended to mitigate the influence of errors, committed on one of the steps, on the other processing step, as well as to determine the bottleneck sub-task in terms of accuracy. We also had an opportunity to test the window approach, proposed by the authors of SENNA and used in NLPNET, for the spatial pivot identification, which is the required initial step.

One of the peculiarities of SpRL, in contrast to SRL, is the possibility that a spatial indicator could serve as the pivot for several spatial relations at a time. The evaluation of the system's performance for the case of static semantic relations shows that this issue is the main hampering factor. Our solution should be capable of treating this difficulty via introducing several changes to the format of the data, that serves as input for the model training script.

Besides relying on a convolutional neural network and the changes introduced to the input data formalism, our approach also builds on using word embeddings, trained in an unsupervised fashion, as the main features.

## 1.3 Contributions

We believe that this work is a demonstration of the fact, that SpRL can be successfully tackled by a system originally designed for SRL. Moreover, certain improvement may be achieved by introducing several changes to the format of the data fed to the training script as input. This is true not only for static spatial relations, but for dynamic relations as well.

Different neural network configuration modifications were tested, with the most suitable having been finally adopted. The main tuned parameters were the number of training rounds and the way that word representations should be initialized. The usefulness of leveraging *word embeddings*, previously trained in an unsupervised fashion, was demonstrated by the improved values of precision and recall.

We hope that our work might constitute a step forward in understanding spatial semantics. Besides being interesting for researches and linguists, Spatial Role Labeling may be used in a number of neighboring fields, that can benefit from the analysis of spatial semantics encoded in natural language. For example, it may be plausible to make use of SpRL achievements for building automated navigation systems, where the path, that a vehicle, a ship or an aircraft should follow, can be encoded in natural language and further on interpreted by the system without human interference. One more interesting application might be geoinformation management systems or different kinds of robots designed for searching objects, that could receive indications on the site to be explored and the relative positions of the objects of interest, expressed in the form of natural language. SpRL is also a key task for text-to-scene conversion, i.e., generation of textual descriptions from visual data [6].

Speaking in more general terms, this work might inspire further research in other NLP tasks, having demonstrated that using neural networks in combination with previously trained word embeddings is a promising issue. It is in line with the modern trend in NLP to reduce the amount of the training corpus, striving to move towards self-learning systems.

## 1.4 Document Outline

The rest of the dissertation is organized as follows: Chapter 2 describes the fundamental concepts used in the area, as well as the previous achievements in the fields of SRL and SpRL (i.e., the existing systems performing Semantic and Spatial Role Labeling). Chapter 3 describes the modifications and the improvements introduced to tackle the SpRL task. Chapter 4 is dedicated to the evaluation methodology and lists the obtained results. Chapter 5 draws the conclusions and also mentions some problems we find interesting for the possible future investigations.

# 2

# Fundamental Concepts and Related Work

## Contents

This chapter reviews the fundamental concepts used throughout the MSc thesis. We begin with formalizing the SRL and the SpRL tasks and further continue by describing the traditional machine learning approaches to sequence tagging, which is a general task that Semantic and Spatial Role Labeling are based on. These approaches are described with respect to the chronological order of their appearance, i.e., from Hidden Markov Models (HMMs) to Conditional Random Fields (CRF).

The second part of the chapter provides an overview of the previously designed systems for SRL and SpRL. We studied some of the existing systems for SRL, namely SENNA, NLPNET, and a high-order low-rank tensor-based system. We also analyzed two of the previously created systems for SpRL, that competed at the SemEval-2012 workshop, namely the KUL and the UTDSpRL systems.

## 2.1 The Spatial Role Labeling Task

Talking about spatial relations is one of the essential functions of natural language. It is often necessary to parse a sentence for determining and classifying the spatial expressions mentioned in it. This intention falls in line with the holistic approach, having utterances, rather than isolated words, as the main unit of analysis [10]. SRL and SpRL are all holistic semantic analysis tasks. The aim of the SpRL task was first defined by analogy with the SRL task, and it assumed the automatic labeling of words or phrases, in a sentence, with a set of spatial roles.

Following the tradition established by the founding fathers of SpRL, let us consider the phrase *Give me the grey book on the big table* as an example. Here we can mark out the trajector *the grey book* (or simply *book*), the landmark *the big table* (or simply *table*) and the spatial relation between the trajector and the landmark, designated by the preposition *on* (which is called a *spatial indicator*, in this case). The basic arguments/roles are, thus, the spatial indicator, the trajector and the landmark. Secondary spatial roles, reflecting direction, distance or motion, can be further introduced.

After defining the spatial roles, the proponents of the SpRL task found that it was necessary to agree upon a common annotation scheme based on the ideas of holistic spatial semantics. Using one such scheme, spatial relations and roles, as well as their indicators and formal meanings, can be tagged in a text. A tokenizer can be used to mark the position of each word in the text. This position is then used as the word's index to address it for the purpose of building annotations. Training corpora for the SpRL task were produced through one such procedure, leveraging also an XML encoding format. These corpora can later be fed to machine learning algorithms.

The trajector is the entity (i.e. the object, person or event) whose location or motion is of relevance. A trajector may be marked out practically in any sentence, with the notable exception of imperative utterances. In the case of sentences like *go right* the implicit trajector is the pronoun *you*.

The landmark is the reference entity in relation to which the location or motion of the trajector is

determined. In some cases it may not be needed (e.g., in the case of sentences mentioning directions, such as *go right*). It may also be deleted by ellipsis and appear implicit (e.g., in the sentence *the balloon went up*) or require resolution in pre-processing or post-processing stages (e.g., in the sentence *the balloon went over there*). Another case is when there is a motion with spatial effect and the landmark is a path rather than just a location (e.g. in the sentence *Mike went out of the room*). This latter problem may be solved either by introducing an additional attribute to the landmark spatial role or by considering an additional spatial role.

The spatial indicator is usually a preposition but may also be a noun, an adverb, a verb or their combination [10]. It is a sort of spatial pivot that is assumed per each relation. The spatial arguments are related to this pivot. Drawing an analogy with the SRL task, we may state that spatial indicators in the SpRL task play a part similar to the one that predicates do in SRL. One challenging problem is to detect the cases of prepositions that usually signify spatial relations being used in other contexts. For example, the sentences *there is a book on the table* and *there is a party on Sunday* have the same structure. However, it is clear that in the latter case the preposition *on* does not indicate a spatial relation, but instead a temporal one.

The motion indicator represents a spatial role conveying the motion of the trajector of a spatial scene along a path [11]. For example, in the sentence *we left Faro for Lisbon* the verb *left* plays the role of the motion indicator.

The path is a spatial role label that is assigned to a word/phrase that denotes a path along which the trajector's motion is performed [11]. It may denote a route of motion (e.g., *along the street*, *on E40*), a starting or ending point of motion (e.g., *he left the office* and *he arrived in Lisbon*, respectively), as well as contain both start and end points of motion (e.g, *she left Lisbon for Oporto*) or a landmark traversed in motion (e.g., *he crossed the bridge*). In complex cases, the phrase, tagged with the path label, may be subject to further semantic parsing to determine the spatial roles in it. For example, the phrase *we drove the villages outside town into Mountain Ridge Park* allows for marking out the trajector (i.e., *we*), the motion indicator (i.e., *drove*) and the path (i.e., *the villages outside town into Mountain Ridge Park*). However, the last phrase may further be parsed to determine the trajector (i.e., *the villages*), the landmark (i.e., *town*) and the spatial indicator (i.e., *outside*).

The direction spatial role denotes an absolute or relative direction of motion, or a spatial arrangement between a trajector and a landmark [11]. For example, in the phrase *the cathedral is located two kilometers west of the palace*, the word *west* serves as a direction indicator.

Finally, the distance spatial role is assigned to a word or a phrase that denotes an absolute or relative distance of motion, or the distance between a trajector and a landmark [11]. In the sentence *the cathedral is located two kilometers west of the palace*, the phrase *two kilometers* should be labeled with a distance tag.

In the SemEval-13 [11] workshop, spatial relations were further classified as either static or dynamic. Static relations are held between a trajector and a landmark. Dynamic relations stand for the usage of a trajector and a path with an explicit or implicit mention of motion, direction or distance. Spatial relations may be further classified into directional, regional and distal. Directional relations are those containing relative directional references (e.g., *the farm is located in the south of the town* or *after leaving the motorway I had to lower the speed*). Regional relations usually mark an absolute position of the trajector with regard to the landmark (e.g., *the car at the paved road* or *the farm outside the town*). Distal relations reflect a scalar entity that can be represented either qualitatively (e.g. *close* or *far*) or quantitatively (e.g., *two meters far*). Distal relations can be further classified into relative (i.e., describing the distance between two entities compared to a third one) and absolute (i.e., describing the distance between two entities without a reference to a third one).

In the SemEval-15 workshop, the SpRL task was incorporated into the Spaceval task [12]. However, the recognition of spatial elements was actually based on the SpRL task formulated at SemEval-13. Participating systems should first identify the spatial entities in a text (such as locations, paths or events), and then classify them according to their type, as well as determine their attributes.

## 2.2 Machine Learning for NLP

Machine learning focuses on designing algorithms for discovering hidden patterns or general rules in the provided data, thus making it possible to make further predictions and decisions on new, previously unseen, data examples.

There are various ways of classifying machine learning approaches. For instance, they can be categorized according to the specificity of the task they are intended to address, namely classification, clustering, density estimation, regression or dimension reduction [13]. A more widely used classification distinguishes between supervised machine learning, unsupervised machine learning and reinforcement learning techniques [14]. In the case of supervised machine learning approaches, the learner is provided with a *training corpus*, i.e., a number of sample inputs together with the corresponding labels signifying the desired outputs. The pursued goal is, thus, to work out a rule that maps the inputs to the outputs correctly. Conversely, other types of learning do not imply labeling of sample inputs: the task is either to discover hidden patterns in unlabeled data (in the case of unsupervised machine learning), or to optimize the specified objective function by directly interacting with the dynamic environment (in the case of reinforcement learning). Since NLP problems started being addressed with the help of machine learning techniques, it has been perceived that supervised machine learning is the approach that yields more accurate results. However, the main tendency among researches in the field is to reduce the size and specificity of the training data, endeavoring to move forwards to self-learning systems [1, 9].

### 2.2.1 POS Tagging as a Simple NLP Task

One of the simplest NLP tasks, where machine learning was introduced, was part-of-speech (POS) tagging. This task belongs to the class of *sequence tagging tasks*. The objective is to build a structured prediction model, whose input is a sentence (i.e., a sequence of words in a natural language), and which produces an output containing a tag sequence, with the tags representing the parts of speech that the elements in the input sequence (the words) belong to. The tag sequence has the same length as the input sequence and, therefore, it specifies a single tag for each word in the sentence. We also have a set of training examples ($x^{(i)}$, $y^{(i)}$), where each $x^{(i)}$ is a sequence of elements (a sentence) ($x_1^{(i)}$, $x_2^{(i)}$,...$x_{n_i}^{(i)}$) and each $y^{(i)}$ is a tag sequence ($y_1^{(i)}$, $y_2^{(i)}$,...$y_{n_i}^{(i)}$). The task is to learn a function that maps word sentences to tag sequences from these training examples, i.e., that maps every input to the appropriate POS label [15].

In fact, many other NLP problems can also be viewed as sequence tagging tasks, with the attributed tags receiving more complex interpretations. A traditional approach, and probably the easiest way to address sequence tagging tasks, is based on the application of hidden Markov models (HMMs) [15].

### 2.2.2 Hidden Markov Models and the Generative Approach

Like any other Markov model, a Hidden Markov Model (HMM) is a stochastic model used to describe randomly changing systems. It possesses the Markov property, which implies that the conditional probability distribution of the future state of the process (conditional on both past and present states) depends on the present state and not on the sequence of the previous events. In case the future state of the system depends not only on the current state, but also on *n* previous states, the model is known as an *n-order Markov model*. A peculiarity of a HMM is that the sequence of states, through which the system passes, is not directly visible. The output sequence, dependent on the hidden states and in some way reflecting them, is what is visible. We can thus conclude that each HMM is characterized by 3 probability distributions: the distribution $\mathrm{P}(y_i)$ over the hidden states, the transitional distribution from one hidden state to another $\mathrm{P}(y_i|y_{i-1}, ..., y_{i-n})$ and, finally, the distribution $\mathrm{P}(x_i|y_i)$ of the observed symbols over the hidden states.

To develop HMMs, the *generative approach* is usually used. This approach focuses on determining the joint probability of observing the input sequence $x = (x_1...x_n)$ paired with the output tag sequence $y = (y_1...y_n)$. This joint probability can be further decomposed in a way that takes its source in the conditional probability definition.

$$\mathrm{P}(x, y) = \mathrm{P}(x|y) \times \mathrm{P}(y) \tag{2.1}$$

It is significant to point out that the final goal of any sequence tagging task is to find the sequence of labels *y* having the highest value of conditional probability $\mathrm{P}(y|x)$ over the input sentence. In other

words, it is necessary to determine:

$$\arg\max_{y \in Y} \mathrm{P}(y|x); \tag{2.2}$$

However, using the Baesian rule, it is not hard to demonstrate, that maximizing $\mathrm{P}(y|x)$ is equivalent to finding the argument $y \in Y$ that maximizes the joint probability $\mathrm{P}(x, y)$. The following formulas justify the usage of generative models for sequences tagging tasks.

$$\arg\max_{y \in Y} \mathrm{P}(y|x) = \arg\max_{y \in Y} \frac{\mathrm{P}(y)\mathrm{P}(x|y)}{\mathrm{P}(x)} = \arg\max_{y \in Y} \mathrm{P}(y)\mathrm{P}(x|y) = \arg\max_{y \in Y} \mathrm{P}(x, y); \tag{2.3}$$

It is also necessary to mention the two main properties of generative models. Firstly, for any combination $(x_1...x_n, y_1...y_n)$ of input and output elements, the following condition should be met:

$$\mathrm{P}(x_1...x_n, y_1...y_n) \geq 0; \tag{2.4}$$

Secondly, for all possible combination of inputs and outputs, the normalization requirement should be held:

$$\Sigma \mathrm{P}(x_1...x_n, y_1...y_n) = 1. \tag{2.5}$$

In accordance with the characteristic property of m-order Markov models, the probability (in the particular case of m = 2) is defined as follows:

$$\begin{gathered} \mathrm{P}(X_1 = x_1, ..., X_n = x_n, Y_1 = y_1, ..., Y_n = y_n) = \\ \prod_{i=1}^{n+1} \mathrm{P}(Y_i = y_i | Y_{i-2} = y_{i-2}, Y_{i-1} = y_{i-1}) \prod_{i=1}^{n} \mathrm{P}(X_i = x_i | Y_i = y_i); \end{gathered} \tag{2.6}$$

We assume that $y_0 = y_{-1} = *$, where '*' is a special start symbol. Further, if we assume that $\mathrm{q}(s|u, v)$ is the probability of seeing the tag $s$ immediately after the sequence of tags $(u, v)$, and $\mathrm{e}(x|s)$ is the probability of seeing observation (an element of the input sequence) $x$ paired with state (tag) $s$, we can provide another form for the joint probability $\mathrm{P}(X_1 = x_1...X_n = x_n, Y_1 = y_1...Y_n = y_n)$ defined above.

$$\begin{gathered} \mathrm{P}(X_1 = x_1...X_n = x_n, Y_1 = y_1...Y_n = y_n) = \mathrm{P}(x_1...x_n, y_1...y_n) = \\ \prod_{i=1}^{n+1} \mathrm{q}(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^{n} \mathrm{e}(x_i | y_i); \end{gathered} \tag{2.7}$$

Let us now focus on a rather intuitive approach to the problem of estimating the parameters of a sequence tagging model, provided that we possess access to the training data. Assuming that we define $\mathrm{c}(u, v, s)$ as the number of times, that the sequence of tags $(u, v, s)$ appears in the training corpus, and $\mathrm{c}(u, v)$ corresponds to the number of occurrences of the sequence $(u, v)$ in the same training data, the

probability $\mathrm{q}(s|u,v)$, defined above, is calculated using the definition of conditional probability:

$$\mathrm{q}(s|u,v) = \frac{\mathrm{c}(u,v,s)}{\mathrm{c}(u,v)} \tag{2.8}$$

The probability $\mathrm{e}(x|s)$, defined above, is calculated using the number of times, that state (tag) $s$ is being matched with the observation $x$ in the corpus $\mathrm{c}(s->x)$, as well as the number of appearances of the state s in the corpus $\mathrm{c}(s)$.

$$\mathrm{e}(x|s) = \frac{\mathrm{c}(s->x)}{\mathrm{c}(s)} \tag{2.9}$$

In what concerns the procedure for estimating the most likely tag sequence for a given input sequence, the most straightforward method, which nonetheless has a very poor performance, consists in enumerating all possible label sequences $(y_1...y_{n+1})$, scoring them and finally taking the sequence with the highest score. The origin of the inefficiency lies in the number of sequences to process: it equals to $|T|^n$, where $|T|$ is the number of different tags to be considered and $n$ corresponds to the input sequence length. A viable alternative to the brute force approach is making use of dynamic programming algorithms. One of the most famous examples of those is the *Viterbi algorithm*, proposed in 1967 [15].

The Viterbi algorithm is based on the fact that the joint probability can be presented in the following form:

$$\mathrm{P}(x_1...x_n, y_1...y_n) = \mathrm{r}(y_1...y_n) \times \mathrm{q}(y_{n+1}|y_{n-1}, y_n) = \mathrm{r}(y_1...y_n) \times \mathrm{q}(STOP|y_{n-1}, y_n) \tag{2.10}$$

In the formula, $\mathrm{r}(y_1...y_k)$ is the probability of observing the tag sequence $(y_1...y_k)$ in case that only the first $k$ members of the input are considered; STOP is a special symbol which always appears at the end of tag sequences. Let $\mathrm{S}(k,u,v)$ be the set of all tag sequences of length $k$ that end in the bigram $(u,v)$, and let $\pi(k,u,v)$ be the maximum of the joint probabilities for the sequence of length $k$, formally defined as:

$$\pi(k,u,v) = \max_{<y_1...y_k> \in \mathrm{S}(k,u,v)} \mathrm{r}(y_1...y_k) \tag{2.11}$$

It is also necessary to define that $\pi(0,*,*)$ = 1 and $\pi(0,u,v)$ = 0, which results from the assumption that $y_0 = y_{-1} = *$. One of the bases of the method is also the recursive definition of $\pi(k,u,v)$.

$$\pi(k,u,v) = \max_{w \in T} (\pi(k-1,u,v) \times \mathrm{q}(v|w,u) \times \mathrm{e}(x_k|v)) \tag{2.12}$$

Here, $T$ is the set of possible tags, and $w = y_{k-2}$ is the $(k-2)$nd element of all the sub-sequences ending in the bigram $(u,v)$. Finally, there is no difficulty in understanding that the sought maximum

probability is expressed by the formula:

$$\max_{y_1 \cdots y_n} \mathrm{P}(x_1 ... x_n, y_1 ... y_n) = \max_{u \in T, v \in T} (\pi(n, u, v) \times \mathrm{q}(STOP|u, v)) \tag{2.13}$$

To determine not only the maximum probability, but also the tag sequence corresponding to that probability, a *backpointers* mechanism should be implemented. The procedure involves storing the backpointer values $\mathrm{bp}(k, u, v)$ at each step, which record the previous state $w$ that leads to the highest scoring sequence of length $k$ ending in $(u, v)$. Once the maximum probability is calculated, it is necessary to perform a return trace and discover the sought highest probability tag sequence, using the stored backpointers.

### 2.2.3 Linear Classification Models and the Discriminative Approach

The generative approach to NLP has some important limitations. For example, in more advanced sequence tagging tasks, such as SRL, the input sequence elements are normally better characterized by a large number of features. There are also complex dependencies between elements of the input vector, that can only be ignored at the cost of reducing performance. In both cases, it becomes very difficult to construct a probability distribution over the input sequence, and new approaches should be sought [16].

A solution is represented by the *discriminative approach*. Here, the conditional probability $\mathrm{P}(y|x)$ is modeled directly, whereas the probability $\mathrm{P}(x)$ is not required. When a new test example is fed to one such model, the most likely label is immediately assigned to the sample [15]. In other words, a discriminative model directly implements the function:

$$\arg\max_{y \in Y} \mathrm{P}(y|x) \tag{2.14}$$

*Linear classification models*, representing the discriminative approach, are also widely used in different NLP tasks. The main objective of these approaches is to determine a *scoring function* for the input $x$, such that:

$$\arg\max_{y \in Y} \Sigma_{i=1}^{N} \theta_i f_i(x, y) \tag{2.15}$$

In the formula, $f_i(x, y)$ is the i-th *feature*, and $\theta_i$ is the corresponding *feature weight*. The classifier further attributes the sample to the class $y$ that has the biggest value of the scoring function.

#### 2.2.3.A The Perceptron Model

The simplest representative of the family of linear classifiers is the *Perceptron*. A standard Perceptron is a binary classifier, which means that it is only capable of separating the samples into two groups.

Multi-class or structured predictions extensions are nonetheless very easy to develop. To perform classification, the Perceptron model first computes the $\mathrm{Z}(x)$ function, such that:

$$\mathrm{Z}(\boldsymbol{X}) = \Sigma_{i=0}^{m} w_i x_i + b \tag{2.16}$$

In the latter formula, $\boldsymbol{X}$ is the input vector, $\boldsymbol{W}$ is the weights vector, and $b$ is the bias weight. The output of the Perceptron is determined by the *activation function*. The *Heaviside step* is used as the activation function in the simplest case, although non-linear functions, such as the sigmoid, are also sometimes used. The Heaviside step may be presented as follows:

$$\mathrm{H}(Z) = \begin{cases} 1, & \text{if } Z \geq C \\ 0, & \text{otherwise} \end{cases} \tag{2.17}$$

Here $C$ is a predefined constant, called the *threshold*.

We say that the data in the training corpus is *separable*, when there exists a weight vector $\theta = (\theta_1, ...\theta_n)$ such that all the data samples can be classified correctly (i.e., the weight vector defines a *separating hyperplane* in this case). On separable data, the training can be performed in a finite number of rounds. Each round is represented by the following iterative procedure [14]:

- The current input sample is taken from the training corpus, and the output is calculated for the sample. A class label is assigned to the sample, according to the output.
- The calculated output is compared to the desired value of the output.
- The elements of the weight vector are updated according to the difference computed at the previous step, so that the model can produce correct predictions.

#### 2.2.3.B Logistic Regression Models and CRF

*Logistic regression models*, used for classification, are a more complex example of linear classification models. Let $\boldsymbol{X} = (x_1, ..., x_n)$ be a features vector, and let $y$ be a binary dependent variable, representing the class label. The logarithm of the probability of each class, given a data sample, is perceived as a linear function of the logarithms of the feature vector elements, plus a normalization constant [16]. Further, the conditional distribution $\mathrm{P}(y|X)$ is given by the following formula:

$$P(y|\boldsymbol{X}) = \frac{\exp(\theta_y + \Sigma_{j=1}^{N} \theta_{y,j} x_j)}{\Sigma_y \exp(\theta_y + \Sigma_{j=1}^{N} \theta_{y,j} x_j)} \tag{2.18}$$

The denominator is a normalization constant, while the value $\theta_y$ is a bias weight, and $\theta_{y,j}$ are weights assigned to each of the $N$ elements of the features sequence.

*Linear-chain Conditional Random Fields models* (hereinafter *CRF models*) can be seen as an extension of logistic regression to structured prediction tasks. They combine the ability of modeling compound outputs and the possibility of taking into account the multiple features of the neighbouring elements of the input sequence. A good *CRF* model is likely to have a simpler structure than an analogous generative model thanks to the fact that the dependencies in the variables of the input sequence, not involving the output, should not be considered [16].

The basic idea of *CRF* models is to attribute a *score* to the possible tag sequences, which are also called *labelings*. This is done leveraging the inventory of *feature functions*. A feature function $\mathrm{f}(x, i, l_i, l_{i-1})$ is a binary function, normally returning the values 0 or 1 (although it can be a real-valued function as well), which, roughly speaking, represents a statement, regarding a pattern in the input data and the conclusion, that can be drawn upon observing that pattern [16]. In a standard CRF, the input parameters of a feature function are: the observed sequence $x = (x_1...x_n)$, the position $i$ of the current element (word) and the tags $l_i$ and $l_{i-1}$ corresponding to the $i$-th and $(i-1)$th elements of the observed sequence respectively. For example, the statement that an English word, ending in *-ly* is an adverb, can be represented by a feature function $\mathrm{f1}(...)$, returning 1 if the said property is the case and 0 otherwise. Another example of a feature function $\mathrm{f2}(...)$ returns 1 if two prepositions go immediately one after the other and 0 otherwise. The next step is to determine the *weights* for each of the feature functions in order to estimate the degree to which the statement, represented by the corresponding feature function, can be considered true. It is intuitively obvious that the feature function $\mathrm{f1}(...)$ should be assigned a high weight, because the statement that it represents is a well known linguistic fact. Conversely, the function $\mathrm{f2}(...)$ should be weighted with a negative value, which would reflect that the underlying linguistic phenomenon is not likely to be observed.

The scoring is performed using the following formula:

$$\mathrm{s}(l|x) = \Sigma_{j=1}^{m}\Sigma_{i=1}^{n}\theta_j \mathrm{f_j}(x, i, l_i, l_{i-1}) \tag{2.19}$$

It is further possible to define the probability of a labeling sequence by exponenting and normalizing:

$$\mathrm{P}(l|x) = \frac{\exp(\mathrm{s}(l|x))}{\Sigma_{l'}\exp(\mathrm{s}(l'|x))} \tag{2.20}$$

From Equation (2.20), one can see that CRF models have much in common with logistical regression models. In fact, the assumption that a single set of weights is shared across all the classes, rather than considering only one weight vector per class, bridges the divide between logistical regression models and CRF. A feature function is basically a variable that equals to 1 for the elements of a single class. Thus, indexing each feature function $f_n$ with its weight $\theta_n$, we get an expression for conditional probabil-

ities in CRF models, where $\mathrm{Z}(x)$ represents the normalization constant:

$$\mathrm{P}(y|X) = \frac{\exp(\Sigma_{j=1}^{N}\theta_n \mathrm{f_n}(y,x))}{\mathrm{Z}(x)} \tag{2.21}$$

To estimate the weights of the feature functions, methods similar to the ones employed for logistic regression parameters estimation are used. The most popular is probably the *maximum likelihood* estimation procedure [16]. The parameters are chosen in a way that they maximize the probability of the training data under the model. In other words, given the training corpus, consisting of sample input sequences ($x_1^{(i)}$, $x_2^{(i)}$,...$x_{n_i}^{(i)}$) and sample output sequences ($y_1^{(i)}$, $y_2^{(i)}$,...$y_{n_i}^{(i)}$), it is necessary to maximize the function $f(\theta)$, such that:

$$f(\theta) = \Sigma_{i=1}^{N} \log(\mathrm{P}(y^{(i)}|x^{(i)},\theta)); \tag{2.22}$$

In linear-chain CRF, the model parameters can be determined with the help of numerical optimization methods, such as *gradient ascent* [16]. At the beginning, the weights are initialized with some random values. Afterwards, the following iterative procedure is executed until the stopping condition is met.

- For each feature function the gradient of the log probability of the training example is calculated:

$$\delta = \frac{\partial \log(\mathrm{P}(l|x))}{\partial \theta_j} = \Sigma_{j=1}^{m}\mathrm{f}(x,j,l_j,l_{j-1}) - \Sigma_{l'}\mathrm{P}(l'|x)\Sigma_{j=1}^{m}\mathrm{f}(x,j,l'_j,l'_{j-1}); \tag{2.23}$$

- The value for $\theta_i$ is updated using a constant $\alpha$ as the *learning rate*):

$$\theta_i = \theta + \alpha\delta; \tag{2.24}$$

Once the model is trained, i.e. its parameters are setup, it is necessary to determine the way in which it is going to predict labels for new input sequences, i.e., to perform *inference*. A version of the Viterbi algorithm, previously described for the case of HMMs, can be used to solve the problem [16].

Being a discriminative generalization of HMMs, CRF models are much more powerful. Firstly, this is due to the possibility of representing more global dependencies between the elements of the sequences, while in an HMM only the probabilities of seeing a word, labeled with a certain tag, and the probability of getting a tag, provided *m* previous tags, can be reflected. Secondly, CRF do not impose any restrictions on the feature functions, while in the case of HMMs it is necessary to ensure the fulfillment of the constraints, so that the parameters $P(x|y)$ can be interpreted as probabilities [16].

#### 2.2.3.C Support Vector Machines

There are also discriminative models which are not probabilistic, namely the aforementioned Perceptron approach. Another important representative here is the *Support Vector Machines (SVM) model*,

which is widely used for binary classification tasks, although extensions for structured predictions also exist. The underlying motivation is that it is often hard to model either the probability $\mathrm{P}(y|x)$ or $\mathrm{P}(x, y)$ directly. Instead, this approach focuses on predicting accurately the outputs according to a defined cost function. An implicit mapping of inputs to higher-dimensional feature spaces can also be used to deal with linearly inseparable data.

To perform binary classification, SVM rely on a function $\mathrm{F}(\boldsymbol{X})$, computed over the given previously unseen input vector $\boldsymbol{X}$.

$$\mathrm{F}(\boldsymbol{X}) = \mathrm{sign}(\boldsymbol{W} \cdot \boldsymbol{X} + b) \tag{2.25}$$

Data samples are classified with respect to the *separating hyperplane*. The weights vector $\boldsymbol{W}$ in the previous formula may be interpreted as the hyperplane's normal vector. $\boldsymbol{W}$ and $b$ are both model parameters to be trained.

In the case of linearly inseparable data, mapping to higher dimensional spaces with a $\Phi(\boldsymbol{X})$ mapping function is used. The classifying function $\mathrm{F}(\boldsymbol{X})$ becomes:

$$\mathrm{F}(\boldsymbol{X}) = \mathrm{sign}(\boldsymbol{W} \cdot \Phi(\boldsymbol{X}) + b) \tag{2.26}$$

The mapping $\Phi(\boldsymbol{X})$ is a *kernel function*, whose main property is:

$$\mathrm{K}(\boldsymbol{X}, \boldsymbol{X'}) = \Phi(\boldsymbol{X}) \cdot \Phi(\boldsymbol{X'}) \tag{2.27}$$

The aim of SVM training is to find the separating hyper-plane that would maximize the distance between objects of separate classes. Let $\rho(y', y)$ be the binary cost function, returning 1 if the label *y'* is erroneously assigned to the data, whose correct label is *y*, and 0 otherwise. The *hinge loss function* $\mathrm{g}(\theta, x, y)$ is defined as follows [14]:

$$\mathrm{g}(\theta, x, y) = \max[\theta(\mathrm{f}(x, y') - \mathrm{f}(x, y)) + \rho(y', y)]; \tag{2.28}$$

For any $(x, y)$ pair, the condition $\mathrm{g}(\theta, x, y) \geq 0$ is met. The hinge loss function equals to 0 if and only if the weights vector provides for a correct prediction. Moreover, SVM models aim at performing confident predictions, with the margin greater than 1, i.e. $\theta\mathrm{f}(x, y') \geq \theta\mathrm{f}(x, y') + 1$ for all $y' \neq y$.

SVM model training can be performed online and carried out in a number of rounds. Each round can be described as follows [14]:

- The current input sample is chosen randomly from the training corpus, and the cost augmented prediction is computed for the current $(x^m, y^m)$ pair.

$$y^{\sim} = \arg\max \theta^t \mathrm{f}(x^m, y') - \theta^t \mathrm{f}(x^m, y^m) + \rho(y', y); \tag{2.29}$$

- The elements of the weight vector are updated using the rule:

$$\theta^{t+1} \leftarrow \arg\max(1 - \lambda\eta_t)\theta^t + \eta_t(\mathrm{f}(x^m, y^m) - \mathrm{f}(x^m, y^{\sim})); \tag{2.30}$$

In the formula, $\lambda$ is a constant, and $\eta_t$ is the learning rate for round *t*.

## 2.3 Related Work

This section surveys previously proposed systems for addressing Semantic Role Labeling (SRL) and Spatial Role Labeling (SpRL).

### 2.3.1 Systems for Semantic Role Labeling

This section describes three state-of-the-art-systems for SRL.

#### 2.3.1.A The SENNA System

The creation of SENNA had the goal of providing a system capable of addressing a vast range of NLP tasks, namely part-of-speech tagging, chunking, named entity recognition and semantic role labeling. In other words, the goal was to avoid *task-specific engineering*, while at the same time achieving state-of-the-art performance in the addressed tasks [1].

SENNA was probably the first system of its kind. To understand the novelty of SENNA, it should be remembered that most of the previous NLP systems were based on linear models with linear kernels and targeted a single NLP problem. Therefore, this class of systems required that researches themselves discover the most proper intermediate representations for task-specific features. These features were often based on linguistic intuition or derived from the output of earlier existing NLP solutions. This approach leads to good performance for separate tasks, but lacks generality in the strive for a broader natural language understanding.

In particular, SENNA is based on a *multilayer neural network architecture*, illustrated by Figure 2.1. Features are learnt step-by-step by successive network layers, with the words of a sentence serving as input for the first layer. The backpropagation algorithm is used to train the discovered features, to ensure their relevance to the task. The authors also argue that it is possible to derive additional benefits from the non-linearity of neural network models, learning more appropriate features for the addressed tasks than with other types of discriminative models. The steps of processing a sentence by SENNA, as well as the functions of each group of network layers, are described next in more detail.

The first layer of the network bears the name of *lookup layer*. Raw words (or strictly speaking *almost* raw words, as some minor preprocessing like lowercasing is still applied) are fed to the neurons of the

lookup layer. For efficiency purposes, the input words are represented by the corresponding embeddings, fetched from a finite dictionary. The aim of the lookup layer is to transform the word occurrences into a feature vector representation, which is done via a lookup table operation. The word *lookup table* is a matrix, whose columns contain vectorial representations of the words. One notable characteristic of the discussed approach is that not only the weights of the inputs of neurons, but also the elements of the lookup table are updated during the training process. The word feature vectors can be thus initialized by random values or using special techniques for extracting word representations from unlabeled corpora [17]. To summarize, assuming that $W$ is the lookup table, and $LT_W(w)$ is the representation of the word $w$, we can state that

$$LT_W(w) =< \boldsymbol{W} >_w \tag{2.31}$$

In the equation, $< \boldsymbol{W} >_w$ is the w-th column of the matrix $W$. The result of applying the lookup operation for a sequence of $T$ words $[w]_1^t$ is a matrix:

$$LT_W([w]_1^t) = (< \boldsymbol{W} >_{[w]_1} < \boldsymbol{W} >_{[w]_2} ... < \boldsymbol{W} >_{[w]_T}) \tag{2.32}$$

It is worth noting that, in a more general case, a word can be represented by a set of features, while each feature is still characterized by a feature vector from the lookup table (i.e., in SENNA, word embeddings can be complemented by additional features, such as POS tags). The user-specified parameter $d_{wrd}^k$ corresponds to the dimension of the parameters vector, associated to the $k$-th feature of word *wrd*. The sentence is still transformed into a matrix of word representation vectors, but these vectors are now a concatenation of feature vectors for all the features, characterizing the word. The matrix has $d_{wrd} \times T$ dimensions, where $d_{wrd} = \Sigma_k d_{wrd}^k$ and $T$ is the number of distinct words in the sentence.

After representing words through feature vectors, the network will assign a tag to each of the input words, i.e. one should take the decision on how the proper tagging task will be addressed. Two alternatives can be used in SENNA: the *window approach* and the *sentence convolution approach*.

The window approach is suitable for less complicated NLP tasks, such as POS tagging, named entity recognition or chunking. The main assumption here is that the word's tag depends only on the *k* words next to it from both sides (i.e., in this case, the hyper-parameter *k* is called the *window size*). As a result of passing all the words of the window through the lookup layer and extracting the feature vectors from the lookup table, the $d_{wrd} \times k$ matrix, corresponding to the window, is formed. The matrix is further transformed into a vector of $d_{wrd} \times k$ dimensions by concatenating all the column vectors. The obtained vector can be fed to the subsequent network layers.

Notice, however, that the window approach is not suitable for complex NLP tasks, such as SRL and SpRL. The problem lies in the fact that the pivot of a semantic relation (i.e., the predicate) may fall outside the window and, therefore, might not be tagged correctly. The *convolutional approach*, aimed at

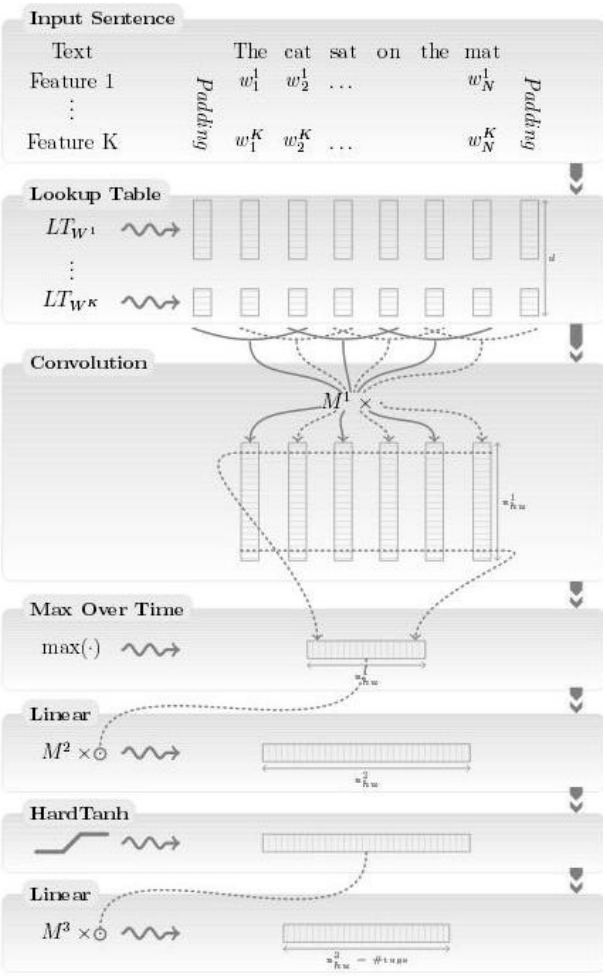


**Figure 2.1:** The convolutional network used in SENNA [1].

considering the whole sentence is deemed useful to tackle the issue.

The feature vectors, serving as the output of the lookup layer, are passed through the *convolutional layer*, which eventually aggregates all the features into a global feature vector, corresponding to the sentence being processed. For each word at position $i$ and for each verb in a sentence (i.e., the possible predicate), the convolutional layer extracts two additional features, representing the relative distance $(i - pos_v)$ with respect to the currently considered pivot, and the distance $(i - pos_w)$ to the word we currently intend to tag. To perform the convolution, the notion of window is used, similarly to that of the window approach. Treating windows as successions of feature vectors, the convolution is applied to each window. It is worth noting that the windows overlap, so that the right flank vector of a window is also the left flank vector of the next window. For the $l^{th}$ convolutional layer, the $t^{th}$ output column, corresponding to the $t^{th}$ window, can be represented as a result of a matrix-vector operation:

$$\boldsymbol{F}_\theta^l = W^l < \boldsymbol{F}_\theta^{l-1} >_t^{d_{win}} + \boldsymbol{b}^l \tag{2.33}$$

In the equation, $W^l$ is the parameters matrix (which is the same for all windows in the sentence), and $< \boldsymbol{F}_\theta^{l-1} >_t^{d_{win}}$ is the vector obtained by concatenating the $d_{win}$ column vectors around the $t^{th}$ output column vector of the window matrix.

$$[< \boldsymbol{F}_\theta^{l-1} >_t^{d_{win}}]^T = (< \boldsymbol{F}_\theta^{l-1} >_{1,t-d_{win}/2} ... < \boldsymbol{F}_\theta^{l-1} >_{d_{wrd},t-d_{win}/2}, ..., \\ ..., < \boldsymbol{F}_\theta^{l-1} >_{1,t+d_{win}/2} ... < \boldsymbol{F}_\theta^{l-1} >_{d_{wrd},t+d_{win}/2}) \tag{2.34}$$

It is worth noting that the vectors, produced as a result of convolution, are variable-sized, with the actual dimension depending on the number of words in the sentence. However, the subsequent network layers are designed to process fixed-size feature vectors. It is thus crucial to eliminate the less important features or apply an average operation to the convoluted feature vectors. In the SENNA system, the former approach was used, which means that given the matrix $F_\theta^{l-1}$, fed to the *max pooling layer* as input, the $i$-th output vector is calculated by the formula:

$$[\boldsymbol{F}_\theta^l]_i = \max_t [F_\theta^{l-1}]_{i,t}, 1 \leq i \leq n_{hu}^{l-1} \tag{2.35}$$

In the equation, $n_{hu}^{l-1}$ is called the *number of hidden units*.

The fixed-sized vector $\boldsymbol{F}_\theta^1$, obtained as the output of the lookup layer in the case of the window approach or of the max pooling layer in the case of the convolutional approach, is then fed to a stack of *linear layers*. Each linear layer applies a transformation to the feature vector, so that:

$$\boldsymbol{F}_\theta^l = W^l \boldsymbol{F}_\theta^{l-1} + \boldsymbol{b}^l \tag{2.36}$$

In the equation, $W^l$ and $b^l$ are the training parameters. The size of the feature vector, output by the linear layers, equals the number of hidden units.

The pack of linear layers is usually interleaved with non-linear layers to consider non-linear features. In those layers, a non-linear function is applied to each element of the feature vector. In the particular case of SENNA, the $\mathrm{HardTanh}$ function is used. The transformation may thus be represented as follows:

$$[\boldsymbol{f}_\theta^l]_i = \mathrm{HardTanh}([\boldsymbol{f}_\theta^{l-1}]_i) \tag{2.37}$$

The HardTanh function can be formally defined as follows:

$$\mathrm{HardTanh}(x) = \begin{cases} -1, & \text{if } x \leq -1 \\ x, & \text{if } -1 \leq x \leq 1 \\ 1, & \text{if } x \geq 1 \end{cases} \tag{2.38}$$

The size of the *output network layer* corresponds to the number of possible tags for the addressed task. The output of the i-th element of the last layer can be seen as the scoring $[\mathrm{f}_\theta(x)]_i$ for the $i$-th tagging option. The discovered scorings are attributed to the word in the center of the window in the case of the window approach, or to the word which is marked as target in the network input in the case of the sentence approach. The scorings may be converted to probabilities by normalization and exponentiation (i.e., the latter is necessary because the scores are interpreted as logarithms). The conditional probability of the $i$-th tag is given by the following formula:

$$\mathrm{P}(i|x,\theta) = \frac{\exp([\mathrm{f}_\theta(x)]_i)}{\Sigma_i \exp([\mathrm{f}_\theta(x)]_i)} \tag{2.39}$$

The logarithm of the conditional probability $\mathrm{P}(y|x,\theta)$ of observing a sample $x$, paired with tag $y$, is then computed.

$$\log \mathrm{P}(y|x,\theta) = [\mathrm{f}_\theta]_y - \log(\Sigma_i \exp([\mathrm{f}_\theta(x)]_i)) \tag{2.40}$$

One particular challenge rests in the fact that simple labeling of separate words by determining the *word-level likelihood* is only appropriate for the POS tagging task. Other NLP tasks, like SRL or chunking, require associating tags to sentence segments. For example, for the SRL task, a chain of words can be tagged as an agent, a cause, a temporal reference, etc. Special *tagging schemes* exist to deal with tagging separate words, taking into consideration their position in the segment. Each labeled word in a segment is tagged with a prefixed label, depending on the word position in the segment (e.g., begin, inside or end). Some minor variations exist in terms of how to attribute the prefixes to the word tags. The SENNA system uses the *IOBES* scheme. The word with tag $X$ receives the *B-X* full tag if it is located in the beginning of a segment, the *I-X* full tag if it is located in the inside of a segment, or the *E-X* full tag if it is located in the end of a segment. A single word with tag $X$ is tagged with tag *S-X*. The words outside the currently labeled segment are tagged with *O*.

There are also tag sequences which are unlikely to appear. To discard such sequences, it is necessary to attribute a score to each tag pair (i.e. define the score of observing one tag after the other). It is thus opportune to consider two matrices. Firstly, $F_\theta(x)$ is the matrix, containing the scores, output by the network (i.e., an element $F[\theta]_{i,t}$ corresponds to the scoring of the $i$-th tag and the $t$-th word of the analyzed sentence). Secondly, let $A$ be the matrix of *transitional scores*, with $A_{i,j}$ signifying the cost of jumping from tag $i$ to tag $j$ in successive words (i.e., $A_{i,0}$ is the score of observing the tag $i$ at the beginning of the sentence). The transition scores are trained, just like all the other parameters of the neural network. In the sentence approach the scoring of a sequence of words $X$ of size $T$ with a tag

sequence $I$ is given by the formula:

$$\mathrm{s}(X, I, \theta) = \Sigma_{t=1}^{T}([A]_{[I]_{t-1},[I]_t} + [f_\theta]_{[I]_t,t}) \tag{2.41}$$

Upon calculating the score, we can perform normalization by dividing it by the sum of the scores for all tagging options, reducing, thus, the score to a probability. The conditional probability $\mathrm{P}(y|x, \theta)$, where $\theta$ is the set of the trained network parameters, and also called the *sentence-level likelihood*, is computed analogously to the separate word approach. As the number of tag paths grows exponentially, the score for all paths, ending in a given tag, should be calculated with the help of dynamic programming techniques, such as the Viterbi algorithm.

To maximize the log likelihood over the training data, a gradient ascent procedure is performed. This implies iterating through the $(x, y)$ samples and making a gradient step:

$$\theta \leftarrow \theta + \alpha \times \frac{\partial \log \mathrm{P}(y|x, \theta)}{\partial \theta} \tag{2.42}$$

After the training process is complete, the system is ready to perform inference, i.e. given a sentence $X$, find a tag sequence $I^*$, such that:

$$I^* = \arg\max \mathrm{s}(X, I, \theta) \tag{2.43}$$

The dynamic algorithms, used to compute the log likelihood of assigning a tag path to the sentence (e.g., the Viterbi algorithm), are again employed to tackle complexity in inference for the sentence approach.

The SENNA system was evaluated by computing *F1* scores for the segments, produced by the model, in different NLP tasks, such as chunking or named entity recognition. The POS tagging task was evaluated through a per-word accuracy metric. Generally speaking, the system proved to yield results which are close to those obtained by well-known benchmark systems, designed to deal exclusively with a certain task of interest. For instance, benchmark systems for SRL (i.e., the most complex of the addressed tasks) achieved an F1 result equal to 77,92, while SENNA produced the F1 score of 55,40 and 70,99 (for the separate word likelihood and the sentence-level likelihood approaches respectively). After several improvements, these scores reached the values of 58,18 and 74,15.

It is thus possible to state that the creation of SENNA was a step forward in the field of NLP. We find the SENNA approach promising for tackling the SpRL task as well, and will therefore use this method as the basis for the proposed work focusing on SpRL.

#### 2.3.1.B The NLPNET System

There are several Python re-implementations of the SENNA system, which can be adopted more easily to deal with the SpRL task. The NLPNET system, created in 2013 by a group of researches from the University of São Paulo, Brazil, is one of them [9]. The authors of NLPNET aimed at further developing the ideas underlying the SENNA system, pursuing also the goal of reducing the computational cost of the algorithm employed. They also intended to address the SRL task for languages other than English. For instance, the evaluation of the NLPNET system has focused on Brazilian Portuguese.

In the case of the SRL task, the text corpus used for training and evaluation was built from Brazilian newspapers. The size of the obtained PropBank-Br corpora was seven times less than that of its English counterpart. Hence, additional efforts needed to be applied to deal with the issue of scarcity of data for effective machine learning.

Word representations, different from the ones used for SENNA, were adopted in the experiments. The authors assumed that a word encoding procedure can be considered good in case it produces similar word representations for words with similar meaning and usage. The similarity here is understood in terms of Euclidian distance or cosine similarity. This characteristic of word representations was achieved by the use of pre-trained *numeric feature vectors*. Each vector had 50 dimensions, with values ranging from -1 to 1.

While words outside the dictionary were assigned random feature vectors to avoid the negative effects of noise, the representations were induced on a separate and very large corpus, by a distributional statistics model, called *Hyperspace Analogue to Language (HAL)* [18]. The idea of HAL is to provide an automated procedure to construct *semantic spaces*, based on word co-occurrences. Semantic spaces are vector spaces with axes corresponding to features, that can characterize an object. The system learns the relevant features (i.e., the axes) and the values for the respective components from text corpora. Upon adopting the usage of HAL, the NLPNET developers reported considerable gains, compared to the results shown by SENNA. Other types of word embeddings can be easily used with NLPNET, such as GloVe embeddings [19] or Word2vec embeddings [17].

The input layers of the network encode the words as a concatenation of the word vector itself, a capitalization vector, and additional features vectors, corresponding to the target and predicate distances (only for convolutional networks).

The SRL process within NLPNET is divided into three steps, namely *predicate identification*, *argument identification* and *classification*. An *argument* here is understood as a group of contiguous words, playing a single semantic role (ex., the agent, the beneficiary etc).

On the initial stage, the word tokens are converted to feature vectors (i.e., tokens are initially represented through their word embeddings). The predicate identification network takes a 7 word window as input and determines whether the word in the middle of the window is a predicate. A simplified version

of the learning procedure is employed, which bears a strong resemblance to the window approach used by the SENNA system (i.e., no convolutional layer is required).

During the argument identification stages, the analyzed phrase is delimited and the argument boundaries are marked. The architecture here is very similar to that of the original SENNA system. The result of the process is a sequence of IOBES prefixes, marking arguments boundaries. Training of the argument identification network is performed using a convolutional backpropagation approach, which is identical to the one described above for the sentence approach in the context of the original SENNA system.

On the third stage, argument classification is performed and the sequence of argument labels is output. The network for this subtask has a number of peculiarities, that distinguish it from the network used by SENNA. Firstly, it classifies *arguments* rather than *words*. Hence, the distance features in the vectors reflect the distance to the target argument, instead of the target word. Secondly, no transition score matrix is used for the arguments sequences. The authors argue that their architecture can prevent the overfitting that may occur due to the moderate size of the training corpus. Training of the network for argument classification is done by backpropagation and is analogous to the procedure used for SENNA. The difference lies in the fact that the arguments (not the whole sentence consisting of words) are input to the model one by one.

The decision of adopting a three-step approach allows for identifying the bottleneck subtask. The evaluation revealed, that the argument identification presented such a bottleneck). The three-step approach also provides for separate training of the networks for each subtask (i.e., many features, learnt for the first subtask, may not be relevant for the second subtask), achieving a performance gain compared with neural network systems based on one-step processing.

The final evaluation of the three-step system in SRL over Portuguese texts (i.e., the method which showed the highest performance) yielded scores of 67.99, 67.04% and 68.97% respectively for F1, recall and precision. This section describes two state-of-the-art-systems for SRL.

#### 2.3.1.C A Tensor-Based Approach to SRL

Tao Lei et al. presented a tensor-based approach to SRL [20]. The main intention of the authors was to avoid the usage of empirical linguistic knowledge for feature discovery and engineering. Moreover, it was desirable to outperform not only the systems built on classical statistical approaches, but also to compete with the later developed systems (e.g., systems like SENNA) based on semantic generalizations of lexical features using neural networks.

Specifically, the proposed system leverages a *tensor-based approach* where the goal is to determine the semantic dependencies for each predicate, belonging to the set of the analyzed semantic pivots $\{p_i\}$ for a sentence $x$. The semantic dependencies may be expressed as tuples $(p_i, a_{i,j}, r_{i,j})$, where $a_{i,j}$ is

the argument and $r_{i,j}$ is the corresponding semantic role label. The tuples can also be viewed as arcs in a *semantic dependency graph*. The goal of the analysis is to obtain a credible semantic parse, i.e., a set of predicted arcs $z_{sem} = \{(p_i, a_{i,j}, r_{i,j})\}$.

The input for the system is a *syntactic dependency tree* $y_{syn}$, produced by a state-of-the-art parser, bearing the name of *RBGParser* [21]. The vertices of the tree correspond to the parts of the sentence that can be marked as the *subject*, the *predicate*, the object, or the *adverbal modifiers*, as well as the prepositions adjacent to them. The semantic parser can be formally described as follows:

$$z^*_{sem} = \arg\max_{z_{sem}} \mathrm{S}_{sem}(x, y_{syn}, z_{sem}) \tag{2.44}$$

In the formula, $\mathrm{S}_{sem}(...)$ is the *scoring function* to be learnt. The scoring function in SRL should reflect high-order interactions between the predicate, the argument paths and the associated role labels. In the described approach, a traditional feature scoring function is combined with a *tensor-based* scoring function.

The traditional feature scoring function is the inner product of a parameters vector and a feature vector. Let $\boldsymbol{W}$ be the parameters vector, and let $\boldsymbol{\Phi}(x, y_{syn}, z_{sem})$ be the integral features vector, subsequently arc-factorized, so that $\boldsymbol{\Phi}(p, a, r)$ is the feature vector for one of the arcs of the semantic paths graph. The scoring function is then given by the following formula:

$$\mathrm{S}_{sem}(x, y_{syn}, z_{sem}) = \boldsymbol{W} \cdot \boldsymbol{\Phi}(x, y_{syn}, z_{sem}) = \Sigma_{(p,a,r)\in z_{sem}} \boldsymbol{W} \cdot \boldsymbol{\Phi}(p, a, r) \tag{2.45}$$

The tensor-based scoring function characterizes each semantic arc $(p, a, r)$. The cross product operation is applied to the atomic feature vectors, associated with the four types of information, namely the predicate token with the local context, the argument token with the local context, the dependency label path that connects the predicate to the argument in the syntactic tree, and, finally, the semantic role of the arc. In the simplest case, the predicate feature vector $\boldsymbol{\Phi}(p)$ and the argument feature vector $\boldsymbol{\Phi}(a)$ are one-hot indicator sequences of length $n$, where $n$ is the vocabulary size. The dependency path feature vector $\boldsymbol{\Phi}(path) \in [0, 1]^m$ and the role label feature vector $\boldsymbol{\Phi}(r) \in [0, 1]^l$ are indicator vectors likewise, with $m$ being the total number of unique paths in the training set, and with $l$ being the number of the possible role labels. The atomic vector can also indicate the word lemma for the predicate, the POS tag, and some characteristics of the neighboring tokens. The cross product of the atomic feature vectors yields a high-order rank-1 tensor in $\mathbb{R}^{n\times n\times m\times l}$. Each entry indicates the combination of four atomic features, appearing in the semantic arc $(p, a, r)$. This cross product captures the whole range of combinations over atomic units.

The semantic arc score for the tensor-based scoring is defined via the inner product operation, analogously to the traditional scoring function. If $\boldsymbol{A}$ is a four-way parameters tensor, the scoring is given

by the following formula:

$$\mathrm{S}_{sem}(x, y_{syn}, z_{sem}) = \mathrm{vec}(\boldsymbol{A}) \cdot \mathrm{vec}(\boldsymbol{\Phi}(p) \otimes \boldsymbol{\Phi}(a) \otimes \boldsymbol{\Phi}(path) \otimes \boldsymbol{\Phi}(r)); \tag{2.46}$$

The parameters $\mathrm{vec}(...)$ correspond to the vector representation of a tensor.

Instead of reducing and pruning possible features concatenations, a *low-rank factorization* of the parameters tensor $\boldsymbol{A}$ is assumed. The parameters tensor can be decomposed into a sum of simple rank-1 components:

$$\boldsymbol{A} = \Sigma_{i=1}^{k} \boldsymbol{P}^{(i)} \otimes \boldsymbol{Q}^{(i)} \otimes \boldsymbol{R}^{(i)} \otimes \boldsymbol{S}^{(i)}; \tag{2.47}$$

In the equation, $\boldsymbol{P}^{(i)}, \boldsymbol{Q}^{(i)}, \boldsymbol{R}^{(i)}$ and $\boldsymbol{S}^{(i)}$ represent the $i$-th row of the respective matrices. $\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{R}^{k \times n}$, $\boldsymbol{R} \in \mathbb{R}^{k \times m}$ and $\boldsymbol{S} \in \mathbb{R}^{k \times l}$ are parameters matrices, and $k$ is a small constant.

The factorization of the parameters tensor has two main advantages. Firstly, no complex tensor operations should be performed (i.e., it is possible to operate with atomic vector projections via $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{S}$ instead). Secondly, the sparsity problem in the feature representations is being alleviated, and the model is capable of grasping high-order feature interactions between atomic units without parameter explosion or overfitting.

The *combined scoring function* consists of the traditional scoring and the tensor scoring components, balanced by the global parameter $\gamma \in [0, 1]$, that is tuned in the development set.

$$\begin{aligned} \mathrm{S}_{sem}(x, y_{syn}, z_{sem}) = \gamma \boldsymbol{W} \cdot \boldsymbol{\Phi}(x, y_{syn}, z_{sem}) + \\ (1-\gamma)\Sigma_{(p,a,r) \in z_{sem}} \Sigma_{i=1}^{k} [\boldsymbol{P}\boldsymbol{\Phi}(p)]_i [\boldsymbol{Q}\boldsymbol{\Phi}(a)]_i [\boldsymbol{R}\boldsymbol{\Phi}(path)]_i [\boldsymbol{S}\boldsymbol{\Phi}(r)]_i; \end{aligned} \tag{2.48}$$

The objective of the *learning* process is to optimize the set of the model parameters $\theta$, where $\theta = \{\boldsymbol{W}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{Q}, \boldsymbol{S}\}$. Various methods were tried for the parameters initialization. The simplest approach is a random initialization. However, an alternative approach, based on the *generalized high-order power method*, was adopted. The most important rank-1 components $\boldsymbol{P}^{(i)}$, $\boldsymbol{Q}^{(i)}$, $\boldsymbol{R}^{(i)}$ and $\boldsymbol{S}^{(i)}$ are obtained incrementally for each $i = 1...k$ one-by-one. The method is actually a generalization of the procedure used to find the largest eigenvectors and eigenvalues of a matrix, applied to the high-order tensor case.

The algorithm input is a sparse tensor $\boldsymbol{T}$, which is formed by putting each parameter weight of the $\boldsymbol{W}$ matrix to the corresponding tensor entry. The current rank number $i$, for which the components $\boldsymbol{P}^{(i)}$, $\boldsymbol{Q}^{(i)}$, $\boldsymbol{R}^{(i)}$ and $\boldsymbol{S}^{(i)}$ are going to be computed, and the fixed rank-1 components, previously calculated for for $j = 1...(i-1)$ $\boldsymbol{P}^{(j)}$, $\boldsymbol{Q}^{(j)}$, $\boldsymbol{R}^{(j)}$ and $\boldsymbol{S}^{(j)}$, are also fed as inputs to the initialization module. Four unit vectors $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{s}$ are randomly initialized, and $T^{'}$ is defined by the following formula:

$$\boldsymbol{T}^{'} = \boldsymbol{T} - \Sigma_j \boldsymbol{P}(j) \otimes \boldsymbol{Q}(j) \otimes \boldsymbol{R}(j) \otimes \boldsymbol{S}(j) \tag{2.49}$$

The following parameter updates are performed iteratively until the $norm$ of the form $||\boldsymbol{s}||_2^2$ converges (see below).

$$\begin{aligned}\boldsymbol{p} &=< \boldsymbol{T}^{'}, -, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s} > \\ \boldsymbol{q} &=< \boldsymbol{T}^{'}, \boldsymbol{p}, -, \boldsymbol{r}, \boldsymbol{s} > \\ r &=< \boldsymbol{T}^{'}, \boldsymbol{p}, \boldsymbol{q}, -, \boldsymbol{s} > \\ \boldsymbol{s} &=< \boldsymbol{T}^{'}, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}, - >\end{aligned} \tag{2.50}$$

In the equations, $\boldsymbol{p} =< \boldsymbol{T}^{'}, -, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s} >$ is equivalent to:

$$\boldsymbol{p}_i = \Sigma_{jkl} \boldsymbol{T}_{ijkl} \boldsymbol{q}_j \boldsymbol{r}_k \boldsymbol{s}_l \tag{2.51}$$

Similarly,

$$\boldsymbol{q}_i = \Sigma_{jkl} \boldsymbol{T}_{ijkl} \boldsymbol{p}_i \boldsymbol{r}_k \boldsymbol{s}_l$$

When the termination condition is fulfilled, the sought values of $\boldsymbol{P}^{(i)}$, $\boldsymbol{Q}^{(i)}$, $\boldsymbol{R}^{(i)}$ and $\boldsymbol{S}^{(i)}$ are obtained from the vectors $\boldsymbol{p}$, $\boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{s}$ respectively.

In what regards the learning process, let $D = \{(\widehat{x}^{(i)}, \widehat{y}_{syn}^{(i)}, \widehat{z}_{sem}^{(i)})\}_{i=1}^{N}$ be the N-sized *training set* used for parameters tuning. The parameters are optimized in a *maximum soft-margin framework*. For each pair $(\widehat{x}^{(i)}, \widehat{y}_{syn}^{(i)})$, it is necessary to separate the correct semantic parse $z_{sem}$ from the incorrect ones. In other words, it is required that:

$$\begin{aligned}&\forall z_{sem} \in Z(\widehat{x}, \widehat{y}_{syn}) : \\ &\mathrm{S}_{sem}(\widehat{x}, \widehat{y}_{syn}, \widehat{z}_{sem}) \geq \mathrm{S}_{sem}(\widehat{x}, \widehat{y}_{syn}, z_{sem}) + \mathrm{cost}(z_{sem}, \widehat{z}_{sem})\end{aligned} \tag{2.52}$$

The function $\mathrm{cost}(z_{sem}, \widehat{z}_{sem})$ represents the structural differences between the right and the wrong parsings. It only equals to zero for correct parsing, i.e. when $(z_{sem} = \widehat{z}_{sem})$. The function is positive otherwise, and it is defined as the sum of the errors: 1.0 is added for a false-positive arc, 2.0 for each negative arc and 0.5 if the role labeling is incorrect, but the corresponding predicate-argument pair is found in both the right and the wrong parses.

The parameters are updated for each of the sentences from the analyzed training corpus. For the parses that violate the constraint given by equation (2.52), it is necessary to find the parse $\widetilde{z}_{sem}$ that is characterized by the maximum violation. In other words:

$$\widetilde{z}_{sem} = \arg\max_{z_{sem}} \mathrm{S}_{sem}(\widehat{x}, \widehat{y}_{syn}, z_{sem}) + \mathrm{cost}(z_{sem}, \widehat{z}_{sem}) \tag{2.53}$$

It is further necessary to find an appropriate parameter update $\Delta\theta$ to compensate the violation. A *joint*

*passive-agressive approach* is used to tackle the issue. The scoring $\mathrm{S}_{sem}(...)$ is a non-linear function, that can be approximated with its first-order Taylor expansion:

$$\mathrm{S}(x, y, z; \theta + \Delta\theta) \approx \mathrm{S}(x, y, z; \theta) + \frac{\partial \mathrm{S}}{\partial \theta} \cdot \Delta\theta \tag{2.54}$$

After defining the *hinge loss* function $\mathrm{loss}(\theta)$, it is possible to get the closed form for the parameters update function $\Delta\theta$.

$$\mathrm{loss}(\theta) = \max\{0, \mathrm{S}_{sem}(\widehat{x}, \widehat{y}_{syn}, \widetilde{z}_{sem}) + \mathrm{cost}(\widehat{z}_{sem}, \widetilde{z}_{sem}) - \mathrm{S}_{sem}(\widehat{x}, \widehat{y}_{syn}\}, \widehat{z}_{sem}) \tag{2.55}$$

$$\Delta\theta = \max\{C, \frac{loss(\theta)}{||g\theta||^2}\}g\theta \tag{2.56}$$

In the equation, $g\theta$ is defined as follows:

$$g\theta = \frac{\partial \mathrm{S}_{sem}}{\partial \theta}(\widehat{x}, \widehat{y}_{syn}, \widehat{z}_{sem}) - \frac{\partial \mathrm{S}_{sem}}{\partial \theta}(\widehat{x}, \widehat{y}_{syn}, \widetilde{z}_{sem}) \tag{2.57}$$

The hyper-parameter $C$ (i.e., the regularization parameter) in the last equation controls the maximum step size of each update. Its role is somehow analogous to the role of the learning rate for other approaches.

For inference, the decoding of SRL is formulated as a bipartite maximum assignment problem, that consists in assigning arguments to semantic roles for each predicate.

For evaluation purposes, the proposed system was compared with the 4 systems that showed the best performance at the CoNLL-2009 shared task [22]. The proposed system did not only perform SRL for English, leveraging the training corpus that was compiled from articles from the Wall Street Journal, but also for other languages, namely for Catalan, Chinese, German and Spanish. The system was also benchmarked against a method, that used distributional word representations for SRL [23].

Following the common practice, the system was evaluated on the basis of an F-score metric. For English, the system managed to outperform all the competitors, achieving the value of 86.58. The difference, in comparison to the results shown by other systems, varied from 3.85 to 0.28 points. The average F-score for SRL, in the 5 languages that were considered, reached the value of 75.77.

### 2.3.2 Systems for Spatial Role Labeling

Spatial Role Labeling (SpRL) is a relatively novel task implying spatial relation and role extraction from *unrestricted* natural language, that was only recently addressed using machine learning techniques. This may be explained by the complexity of spatial semantics from the cognitive-linguistic point

of view, by the variety of formal representation models and by the difficulties in coordinating generally accepted specifications [24]. As a result, it was difficult to agree on a common annotation scheme and, thus, create a corpus of annotated data, on which machine learning could have been employed.

The first systems addressing the SpRL task were based on the ideas developed by Parisa Kordjamshidi et al. [7, 8, 10]. The competitors, that appeared at the SemEval-12 workshop and were evaluated on the same data, were the KUL-SKIP-CHAIN-CRF system (hereinafter the KUL system) and the two modifications of the UTDSpRL system [6, 24].

#### 2.3.2.A The KUL System

The KUL system represents a *pipelining approach* to SpRL [7]. First, parts of the input sentence that play the spatial pivot role are identified, leveraging the annotated data of the preposition project [25], which was introduced at the SemEval-2007 conference [26]. The result (i.e., the sentence with the spatial pivot labeled) is fed to the *spatial role labeler*, and the spatial arguments (i.e., the trajector and the landmark) are then identified.

The set of the attributed spatial roles is limited to $\{spatial_indicator, trajector, landmark, none\}$. A *semantic parse tree* serves as auxiliary input data for the task. Notice, however, that in the approach used by KUL, the spatial argument roles are assigned to separate words, rather than to phrases, bearing the respective semantics. In other words, only the *head word* of a phrase is considered, while its modifiers are tagged as *none*. For example, if the phrase *the big table* is a landmark, only the word *table* will be tagged with the respective label, while the modifier *big* will receive the *none* tag.

The spatial relations associated with a sentence, may be defined as follows:

$$SR = \{< w, w', w'' > | w \in I, w' \in T_w, w'' \in L_w\} \tag{2.58}$$

Let $\mathrm{I(w)}$ be a binary function, indicating whether a word $w$, that is a member of the set $I$, is a spatial pivot or not. Analogously, the binary function $\mathrm{T}_w(w')$ shows whether $w'$ is a trajector, provided that $w$ is a spatial indicator (i.e., it can be seen as the characteristic function of the $T_w$ set). Finally, the characteristic function $\mathrm{L}_w(w'')$ reflects whether $w''$ is a landmark, when $w$ has already been tagged as a spatial pivot.

The spatial pivot identification subtask consists in learning the function $\widehat{\mathrm{I}}(x|w, \mathrm{f}(w, S))$, representing the probability of the word $w$ getting the label $x$, where $x \in \{spatial, nonspatial\}$, and where $\mathrm{f}(w, S)$ reflects a set of features. The aforementioned indicator function $\mathrm{I(w)}$ can be now defined as follows:

$$\mathrm{I}(w) = \begin{cases} 1, & \text{if } spatial = \arg\max_{x \in r} \widehat{\mathrm{I}}(x|w, \mathrm{f}(w, S)) \\ 0, & \text{otherwise} \end{cases} \tag{2.59}$$

Using linguistic intuition, it was assumed that only prepositions can serve as spatial indicators, i.e., the $I$ set contains exclusively prepositions. The function $\widehat{\mathrm{I}}(x|w, \mathrm{f}(w, S))$ leverages a number of linguistically motivated preposition contextual features for each preposition $w$ in the sentence $S$, reflected by the function $\mathrm{f}(w, S)$. These features include the preposition itself, the words directly dependent on the preposition, called the *syntactic head1*, and also the words, on which the preposition is directly dependent, called the *syntactic head2*. These latter two groups of features, are obtained by exploiting a *dependency parser*. If the parser detects a relation between the preposition and a predicate, a semantic role labeler is additionally involved for defining the arguments of that predicate, other than the preposition. It is also worth noting that the extracted words are complemented with additional features, such as the lemma, the POS tag, the type of the dependency relation and the semantic role label.

Both a discriminative and a generative classifier (a logistic regression and a *Naive Bayes* classifier, respectively) were tested for the spatial pivot identification subtask.

The trajector and landmark identification sub-task consists in assigning a class label $x$ to the word $w$, where $x \in \{trajector, landmark, none\}$. The authors also assumed that the spatial indicator $s$ has already been marked out in the sentence $S$. The probabilistic classification function $\widehat{\mathrm{R}}(x|w, \mathrm{f}(w, s, S))$ is learnt and maximized afterwards, determining, thus, the most probable label $r_{w,s}$.

$$r_{w,s} = \arg\max_{x \in r'} \widehat{\mathrm{R}}(x|w, \mathrm{f}(w, s, S)) \tag{2.60}$$

It is therefore possible to define the trajector and the landmark characteristic functions $\mathrm{T}_s(w)$ and $\mathrm{L}_s(w)$ for the word $w$, given the spatial pivot $s$, as follows:

$$\mathrm{T}_s(w) = \begin{cases} 1, & \text{if } r_{w,s} = trajector \\ 0, & \text{otherwise} \end{cases} \tag{2.61}$$

$$\mathrm{L}_s(w) = \begin{cases} 1, & \text{if } r_{w,s} = landmark \\ 0, & \text{otherwise} \end{cases} \tag{2.62}$$

The function $\mathrm{f}(w, s, S)$, passed to the classifier $\widehat{\mathrm{R}}(...)$, represents a set of features, defined for the word $w$ in the sentence $S$, provided the spatial indicator $s$. The authors distinguish between three types of features, described by three separate feature functions. The $\mathrm{f}_1(w)$ function is related to the classified word, the $\mathrm{f}_2(s)$ function characterizes the spatial indicator, to which the word is probably related, and, finally, the $\mathrm{f}_3(w, s)$ function describes the features, relating the word to the sentence's indicator.

The first set of features includes the form of the word, the POS tag, the dependency to the *syntactic head* in the dependency tree, the semantic role and the corresponding *subcategorization* (i.e., the *sister nodes* of the *parent node* in the dependency tree).

The features of the spatial indicator comprise the respective word itself and its subcategorization.

The features of the third group contain the path in the parse tree from the word $w$ to the semantic pivot $s$, as well as the position of the word $w$ relatively to the semantic pivot $s$ expressed by a binary variable whose value is *true*, if $w$ is before $s$ and *false* elsewise. The relative distance between the word $w$ and the pivot $s$, which is obtained as the result of dividing the number of nodes between $w$ and $s$ by the total number of nodes in the parse tree. The value of the quotient is rounded, and the reciprocal value is finally included in the feature set.

In the initial experiments, the words, represented by their feature vectors, were classified independently from each other. However, this assumption was not considered viable in practice, as the class, assigned to the word, is likely to constrain the possible attributions for the following word. A CRF model, capable of taking into account complex and long-distance inter-dependencies between words, was deemed useful to tackle the argument classification subtask.

After all the elements of spatial relations were identified, it is necessary to gather them all into spatial relation triplets. This phase implies no machine learning and is rather straightforward.

The pipelining approach is not the only way to address the SpRL task. The authors of the KUL system also experimented with a *joint learning* of all the spatial roles in one step, omitting thus the phase of using a separate classifier for spatial indicators recognition. The latter approach does not provide for exploiting the annotated data source of the preposition project, and therefore suffers from the lack of lexical information. However, theoretically, it allows for some new opportunities, namely the possibility to lift the constraint of spatial indicators being exclusively prepositions, as well as the avoidance of propagating of errors, committed on the stage of spatial pivot identification.

A problem with SpRL lies in the fact that an unlabeled sentence, fed to the model, can contain multiple spatial pivots, as well as several trajectors and landmarks, while the classifier is only capable of attributing a single label to each word. This problem does not arise in the pipelining approach, because it implied processing of only one spatial relation (i.e., a spatial pivot with its two arguments) at a time. The proposed solution requires generating multiple examples from the processed sentence with explicit marking of the spatial pivots and the specific features, extracted for each word. A potential difficulty lies in the theoretical necessity to present as many examples as the number of words in the sentence. However, the authors argue that the approach provides for classifying trajectors, landmarks and spatial indicators in the context of one another, while the number of examples may still be limited to the number of the pivots, that are prepositions. The fact that more than one word can be tagged as a spatial indicator in one sequence can potentially lead to extracting erroneous relations. Meanwhile, the authors noted that such relations are rarely extracted, as each example contains only one preposition.

The KUL system showed better results for the pipelined approach. The authors came to the conclusion that the joint approach led to an increase in the sparsity and overfitting to the training data. Actually,

it was the pipeline version of the system that participated in the SemEval-2012 shared task. The experiments revealed that the system was capable of performing SpRL with reasonable accuracy, although the accuracy and other performance scores were still rather domain-dependent [24]. The final F1 score, obtained for the SemEval-2012 shared task, equaled to 0.5.

#### 2.3.2.B The UTDSpRL System

The UTDSpRL system was conceived as a competitor to the pipelined approach used in the aforementioned KUL system [8]. According to the authors of UTDSpRL, the latter approach has a couple of weak points, resulting from the partitioning of the SpRL process into three separate phases. It is argued that prepositional attachment should be addressed considering the semantic perspective, and each of the two main spatial arguments (i.e., the trajector and the landmark) cannot be identified without additional knowledge about the other one. To prove this statement, the sentence *there is a picture on the wall above the bed* is used as an example. The authors claim that with the implemented approach the triple that is the most likely to be identified is *wall-above-bed*, which contradicts to common sense semantics.

The main idea behind the UTDSpRL system consists in a *joint approach* that decides whether a candidate triplet (i.e., $trajector$, $spatial\_indicator$ and $landmark$) actually reflects a valid spatial relation or not. High-recall heuristics are employed to tackle the issue of recognizing the objects capable of taking part in spatial relations or serving as spatial pivots. All possible combinations of arguments are then fed to a binary classifier, which makes a joint decision.

In the context of *relation candidate selection*, potential spatial indicators are chosen from a closed set of prepositions and some phrases with spatial meaning. A relatively simple lexicon, built from the indicators in the training data, is employed. The indicators, absent in the training corpus can not be detected and constitute a problem of the approach behind the UTDSpRL system.

Conversely, trajectors and landmarks can not be taken from a closed class of objects. In the UTDSpRL heuristic approach, *noun phrase heads* are treated as spatial objects, and three sources are incorporated, namely the NP-heads from the syntactic parse tree, the NP-heads from the chunking result and, finally, the words that are annotated as nouns in 66% of cases in the training corpus. These three heuristics aim at reducing drastically the number of negative instances, i.e., boosting a high recall value.

At the end of the relation candidate selection stage, the discovered spatial pivots and spatial objects are combined to form all possible variants. Besides that, an additional candidate relation with an undefined landmark is formed for each candidate pair.

The subsequent steps include solving a binary classification problem for understanding, which of the obtained candidates are relevant spatial relations. The LibLINEAR SVM classifier is used [27], and

the model parameters are tuned in a way to sacrifice precision for recall, at the same time maintaining a good level on the F1 score (i.e., the weight of the negative outcome is decreased from 1.0 to 0.8). The same SVM classifier, although with the assumption of multi-class classification, is used for the more advanced task of identifying the spatial relation type, distinguishing between directional, regional and distal relations. No weights adjustments are made on this step, and the primary goal is to ensure maximum accuracy.

The system uses a large number of manually engineered features and determines the most important ones via a greedy forward/backward automated feature selection technique. The method iterates through the unused features and chooses the most relevant ones, adding them to the feature set. At the end of each iteration, features that became redundant after adding new features, are removed from the set.

After the feature selection procedure, two main types of features are used at the classification phase: argument-specific features and joint features, concerning two or more of the spatial arguments. Examples of features, belonging to the first group, are the trajector's, the landmark's and the spatial indicator's raw strings, the trajector's and the landmark's lemmas and the last word of the spatial indicator, which is a feature used for determining the relation type. Examples of features of the second type include the path from the trajector to the spatial indicator, output by using a dependency parser. Notice that, in case the landmark of a spatial relation is undefined, the value of the features, related to the landmark, equals to *undefined*.

Two separate submissions were presented at the SemEval-2012 shared task. These submissions differed mainly in some of the used features. More specifically, the first variant of the system (UTDSpRL-SUPERVISED1) employed an additional joint feature based on the conjunctions, such as *and* and *but*. This feature was a concatenation of the path from the trajector to the spatial indicator, the proper indicator raw string and the dependency path from the spatial indicator to the landmark. Notice that all the paths, related to the trajector being considered, were included into the feature set. This feature was removed in the second submission (UTDSpRL-SUPERVISED2) due to the difficulty of detecting conjunctions with high precision.

The evaluation results showed that the second submission had a better performance, achieving the F1 score of 0.573 for relation detection. Both submissions managed to outperform the KUL system, whose F1 result in the competition was only of 0.5.

## 2.4 Overview

In this chapter, we made a survey of the main concepts that are crucial for understanding the idea of this work. We also traced the development of machine learning techniques that could be applied to

sequence tagging tasks of different complexity levels. We explored two classes of approaches to NLP, namely the generative and the discriminative approaches, and explained the differences between them. We also dwelt on a range of examples, including Hidden Markov Models (HMMs), Perceptron, Linear Conditional Random Fields (CRF) and Support Vector Machines (SVM), each of which can be seen as a representative of the two main classes of approaches to NLP.

We studied some of the existing systems for SRL, namely SENNA, NLPNET, and a high-order low-rank tensor-based system. The first two of the mentioned solutions are quite similar, as they build on a convolutional neuronal network approach. Besides that, we analyzed two of the previously created systems for SpRL, having competed at the SemEval-2012 workshop. One of those (KUL) makes use of a CRF model, and the other one (UTDSpRL) uses an SVM to extract spatial relations triplets out from the candidates, which, by-turn, are discovered with the help of a high-recall heuristics.

The approach we advocate is different both from the traditional methods used for sequence tagging and from the approaches, previously employed for SpRL. It is based on the ideas that resulted in the creation of SENNA and NLPNET - the two systems, that have proved their effectiveness in terms of tackling the SRL task.

# 3

# Modifications and Improvements for Spatial Role Labeling

## Contents

This chapter reviews the main features of the network modifications used, as well as the main stages of the work performed, namely the input data format conversion, the system training and the input data transformation to boost the values of the evaluation metrics.

## 3.1 Network Architecture

Spatial Role Labeling is a complex task, which requires several steps. Each of these steps implies machine learning and results in a distinct model, based on a multilayer neural network. To perform SpRL successfully, it is necessary to start by marking out pivots of the spatial relations (i.e., spatial indicators in the case of static relations and motion indicators in the case of dynamic relations). After that, it comes plausible to find the spatial arguments for each of the previously determined spatial pivots and assign the correct spatial roles to each argument. Taking into consideration the importance of feature vectors, updated at training together with many other network parameters, it should be pointed out that the vectorial representations of words should be initialized before starting either of the three mentioned training processes.

Spatial pivot identification is a relatively simple task, that implies determining whether each of the words in the sentence is a spatial pivot or not. It is worth noting that this task may be seen as a light version of part-of-speech (POS) tagging, as spatial pivots are usually prepositions in the case of static relations (although not all prepositions are spatial indicators) and prepositions and verbs in the case of dynamic relations (although some notable exceptions to this rule exist). The little number of options (i.e., only two options are possible) and strong resemblance to the POS tagging task provide for making use of the tools that have proved their efficiency for leveraging POS tagging. More specifically, a SENNA-style window approach with a Multilayer Perceptron (MLP) neural network is employed. The phrases are treated as sequences of 7-sized windows, although NLPNET allows the researcher to set the window size manually. The word, that is currently being tagged, is in the middle of each window.

There are actually two ways of carrying out the rest of the SpRL process. On the one hand, it can be performed in one step, which is the way adopted by the SENNA system for Semantic Role Labeling [1]. On the other hand, two distinct models can be created for *argument identification* and *argument classification*. The first one would aim at determining the spans of the subordinate arguments for every spatial pivot, while the second would attribute the appropriate tag to each of the discovered spans. The latter approach was recommended by the authors of NLPNET, although the system still permits to perform argument identification and argument classification in one step. It is worth noting that in the case of leveraging two separate models for argument identification and classification, the neural networks are quite similar, although certain minor differences, that make both models more task-specialized still exist. Both are *convolutional networks* relying on feature vectors updated in the course

of the training process. This architecture is suitable for complex tasks such as SpRL, because it provides for considering features, related to the whole phrase and not only to the $N$ neighbors of the word being tagged, which would be typical for the window approach. The networks perform *convolution*, i.e., the *convolutional neuron layer* applies a weighted sum to the feature vectors of the overlapping windows. A K-window consists of the feature vector of the word currently being tagged concatenated with the feature vectors of the K neighboring words. Each of the *convolutional neurons* stores the result for every input word. Afterwards, each of the neurons outputs the highest of those values. The results are passed through a hidden layer and afterwards through a non-linear sigmoid function layer, being consequently fed to a linear MLP layer to learn more high-level features. Note that the default number of convolutional neurons equals to 150, but this network can be tuned manualy by the user.

One big advantage of using two distinct neural networks for argument identification and argument classification, besides the opportunity of having a more highly-specialized network for each of the sub-tasks, is that it provides for using features that are mostly appropriate for each of the sub-tasks. It is thus possible to avoid that features, that are relevant for the other subtask, interfere into the training process for a particular sub-task.

Argument identification consists in tagging each word in the phrase with one of the 5 *IOBES tags* related to each of the previously discovered spatial pivots. A word can be tagged as the beginning of a single spatial role span (i.e., *B*), the last word of a span (i.e., *E*), a single word being a role span in its own (i.e., *S*). A word can also be tagged as being within a span (i.e., *I*) or out of any span (i.e., *O*). More specifically, the output of the network for every word is the score, relative to each of the possible *IOBES* tagging option.

Technically, the network for argument classification is analogous to the one used for argument boundaries identification. However, the process has a slightly different interpretation compared to the case of argument identification. Actually, on this step, the previously determined spatial role spans, and not the words, are tagged with one of the possible role labels. The network outputs the scores of every tagging option for each of the spans. One peculiarity of the network used for argument classification is the absence of the hidden layer after the convolutional layer, i.e., a non-linear function is applied directly to the output of a convolutional layer.

It is crucial to remember that for each of the input words there should be an indication of the word or the spatial role span of tokens, that is being currently tagged. The pivot being currently considered should also be indicated. The *distance features* are auxiliary features, aimed to tackle the issue. The tokens within the currently tagged word or role span are assigned distance zero, the tokens, that are to the left of the currently tagged word span, are assigned their relative positions with respect to the most left token of the span. The tokens to the right of the currently considered span are assigned their relative positions with respect to the most right token of the span. The distance features are updated using

*gradient descent* in the course of training, just like all the other network parameters (e.g., the transitional network weights and the word representations). The system allows to set manually the number of the used distance features, with the default value equal to 5.

At the inference phase, our solution leverages the scores, attributed to each of the tagging options for every word, and runs the Viterbi dynamic algorithm procedure to determine the best tagging sequence.

## 3.2 Parameter Tuning for the Training Phase

The *system training* stage was carried out separately for three distinct models, with each model possessing its own neural network. The decision of training the models for *spatial pivot identification*, *argument identification* and *argument classification* separately allows for a finer tuning of the parameters of the training process for each of the stages, making use of the lessons previously learnt by other researches as the result of their experiments with the system.

The most important parameters of the NLPNET training script are the number of rounds, the learning rate and the indication on how the word embeddings should be initialized. Taking into consideration the recommendations of the system's creators and the fundamental similarity of SpRL and SRL, we set the number of rounds equal to 1 for spatial pivot identification, 15 for argument identification and 13 for argument classification. This choice, although motivated largely empirically, has some justifications. Note that although training, performed in an increased number of rounds usually provides for getting a model that is better in terms of accuracy, it may sometimes lead to a considerable diminishing of the error rate decay gradient (i.e., the error rate obtained after successive training rounds tends to remain constant) and to increased time costs.

Spatial pivot identification is the most straightforward task of the three. The tagging of spatial pivots is actually equivalent to tagging prepositions in the case of static role labeling and to marking out verbs and/or prepositions in the case of dynamic role labeling, although it's necessary to remember that not all prepositions are spatial pivots and not all verbs are motion indicators. The most notable example of verbs not acting as a motion pivot are auxiliary verbs. Training of the models for pivot identification was performed in 1 round. The learning rate was set to 0.001.

The argument identification phase is the one that requires an increased number of training rounds due to a relatively large number of the possible IOBES tags and the likely numerous spatial pivots to consider. It was performed in 15 successive rounds with the learning rate set to 0.001.

Training at the argument classification phase was performed in 3 rounds with the learning rate equal to 0.01. After that, 10 more rounds with the learning rate of 0.001 were executed. The learning rate was decreased 10 times in order to prevent overfitting.

No previously trained *word embeddings* were used at the first stage of the experiments, as NLPNET

provides for random initialization of the feature vectors at the training phase. However, due to the relatively small size of the training corpus, the resulting models showed poor performance, so that the experiments with such models were dismissed.

On the contrary, loading the word embeddings, previously trained in an unsupervised fashion, provided for a notable boost in the system performance, with the *precision* and *recall* being the metrics affected. The multidimensional embeddings, distributed on the word2vec website, was the ultimately adopted type of embeddings due to their accessibility, robustness and the great size of the text corpus used to produce them.

The vectorial representations for the words are stored in a binary file and were induced, with the help of the word2vec tool [17], on a large corpus in the English language, provided by Google. Each word is matched with a high-dimensional numerical feature vector. Although the values of the vectors' components seem to be meaningless per se, the comparison of the distances between vectors in terms of *cosine similarity* can draw us to some remarkable conclusions. Not only do the words signifying semantically related concepts have similar representations, but it also appears possible to encode some facts of human knowledge by means of word embeddings.

To be used by NLPNET, the word embeddings should be obligatory represented by two files, stored in the model directory. One of them should be a text file, containing the complete list of words, for which the embeddings vectors were generated. The other one must be a *.npy* file, that contains the values of the vector representations for the words in a format, that is handy for being loaded at the training stage. The system did not previously have built-in functionality for converting the binary word2vec embeddings to the training-friendly format. However, it appeared plausible to solve the problem by leveraging mechanisms similar to the ones used by the system script for other embeddings storage formats (ex. Glove or plain text).

## 3.3 System Training Input Data Format

To carry out system training, the input data should adhere to a certain formalism. In the case of NLPNET, it is required that the CoNLL-2005 shared task format specifications and conventions have to be met.

Both of the gold annotation sub-corpora, used for system training, were obtained in the SemEval-2013 shared task format [11]. The proceedings of this event stipulate that the annotations should be encapsulated in XML format files. An annotation file should contain one XML element, embracing the text, used for training, as well as a set of XML elements, corresponding to the spatial roles and relations observed in the text. These instances are marked out with the use of a semi-automated procedure. The supported basic spatial roles in the case of static relations are the trajector, the landmark and the spatial

indicator. In the case of dynamic relations, this set is enlarged by the motion indicator, the path, the direction and the distance. It is good to remember, that the aforementioned set constitutes the bunch of *basic* spatial roles. Later on, in the course of this work, additional *surrogate roles* were introduced in order to circumvent some of the limitations, that NLPNET has due to having been designed for dealing with SRL instead of SpRL. Each element, matched with a peace of text playing a particular spatial role, should contain references to the start and the end positions of the span in the original text and a unique role identifier, which are stored as attributes of the element. The elements, corresponding to the marked out spatial relations, contain unique identifiers of the roles, that the tuple is formed of. These references are stored as attributes.

Data files that stick to the CoNLL annotation scheme contain sentences, separated by a blank line. Each token that is part of a sentence is characterized by an ordered predefined set of features. The incomplete list of features includes the unique token identifier, the word form that was observed in the currently analyzed sentence, the lemma, the SpRL tag, the unordered set of syntactic or morphological features for the token and the syntactic head of the current token. Note that the presence of syntactic and morphological features is optional, which may be seen as a proof of the organizers' intentions to move away from linguistically motivated feature engineering towards more general and universal approaches to NLP problems. The enumeration of feature values, corresponding to each word token, should start with a new line, and these values should be separated by a TAB symbol.

It was thus necessary to create a script for converting from the XML format, used at the SemEval challenges, to the CoNLL scheme. The programming process was complicated by several inconsistencies in the input data, one example of such being the fact that some tuples contained references to elements, that did not actually correspond to any span within the analyzed phrase. In this case, it was necessary to delete the references to such elements from the tuples, as these ”fantom” roles could never be represented by the CoNLL formalism.

## 3.4 Input Data Format Transformations

We further describe the measures that were taken to circumvent the fact that NLPNET was not originally designed for SpRL, nor was the SpRL formalism meant to be used by a system like NLPNET. Several problems were detected in the formalism that the training data stuck to. It was thus proposed to introduce certain alterations to the basic SpRL formalism, which affected both the case of static and dynamic relations.

### 3.4.1 Static Relations

The survey of the results, obtained for the static relations after having trained the basic version of the model, revealed that the main bottleneck was represented by the phrases, containing multiple spatial relations having the same spatial pivot. The system was capable of discovering only one of the bunch of such tuples, leading to a moderate value of recall. Note, that no similar problem arouse in the case of SRL, so that NLPNET could not deal with such cases without certain alterations being introduced in the system code or the input data format.

The decision on how to treat the mentioned difficulty was strongly influenced by the ripeness of NLPNET as a system, as well as its optimized nature for tackling the SRL task and, as a consequence, its fundamental ability to address the SpRL task with getting tolerable results. Hence, it was found undesirable to introduce changes to the internal structure of the system, opting for focusing on how to work out an input data format transformation, that would help to overcome this issue.

Starting from the fact, that the course of inefficiency rested in that the system assumed each spatial indicator to be a pivot of only one relation, the idea that appeared most natural was to create one *compound relation* out of various relations, having the spatial pivot in common. It was thus necessary to represent these new structures in a *modified annotation file*, that would stick to the SemEval format and could be further converted to the CoNLL style format and be easily used for system training. It was perceived that after the system was trained to learn the relations from the *modified annotations file*, it would be able to produce the desired output at the inference phase, which could be later converted back to the original format (i.e., getting rid of the *compound relations* and producing various simple tuples out of a compound one). The obtained original format file would be treated as the system-produced output at the evaluation stage.

The *compound relations* of the *modified annotation* could contain various roles of the same class (e.g., two trajectors and/or two landmarks). However, the presence of such relations in the annotation used for system training would lead to an ambiguity as NLPNET would never be capable to define at the inference phase, how many trajectors or landmarks the *compound relation* being tagged actually contained. It was thus still necessary to set the maximum number of trajectors or landmarks in a *compound relation* explicitly or to solve the problem by introducing additional *surrogate spatial role classes*. From a formal point of view, the latter option would lead to the introduction of new spatial arguments. From than on, a spatial pivot could not only have had trajectors and landmarks, but also *co-trajectors* and *co-landmarks* serving as its arguments.

The discussed problem was ultimately solved by introducing additional arguments to the spatial relations. Consequently, new spatial role classes were introduced. *Trajector-1*, *trajector-2*, *trajector-3* and *landmark-1*, *landmark-2*, *landmark-3* etc. were the *surrogate spatial role classes* that were assigned to *co-trajectors* and *co-landmarks* of the *compound relations*. Note that the number of new *surrogate*

*classes* was not limited and depended exclusively on what was seen in the training corpus.

A *compound relation*, therefore, comprised the pivot and the arguments that used to represent the *common part* of the aggregated simple spatial tuples, while the rest of the arguments corresponded to the distinct parts of the simple tuples joined together and belonged to a *surrogate role* class.

To illustrate the transformation, let us consider the phrase *Behind it a bench and two people* which is a truncated version of a sentence, observed in the training corpus. According to the tagging rules, that the original annotation sticks to, the spatial relations that should be marked out and consequently learnt by the system at the training stage are $behind(bench, it)$ and $behind(people, it)$. In the first relation, the trajector is $bench$, and the landmark is $it$. In the second relation, $people$ is the trajector, and $it$ again serves as the landmark. We can also observe that $behind$ is the common spatial indicator for both relations, which provides for applying the discussed transformation. After doing that, we get one *compound relation* out of the two simple ones, that will be further included in the modified annotation file. The obtained *compound spatial tuple* is $behind(bench, people, it)$, where $bench$ has the label of *trajector-1*, $people$ is the *trajector-2*, $it$ is the *landmark* and $behind$ is the spatial indicator. Note, that in order to avoid dealing with surplus complexity, there is no need to change the labels assigned to the last two words, as they present the *common part* of the two simple relations.

Similarly, the phrase *There is a chair and a bed next to the table and the wardrobe*, that is represented by the tuples $next\_to(chair, table)$, $next\_to(bed, table)$, $next\_to(chair, wardrobe)$ and $next\_to(bed, wardrobe)$, results in a *compound relation* $next\_to(chair, bed, table, wardrobe)$ in the *modified annotation* file. In this *compound relation*, *chair* is the *trajector-1*, *bed* is the *trajector-2*, *table* is the *landmark-1* and *wardrobe* is the *landmark-2*, while *next to* is still a *spatial indicator*, being the *common part* of the 4 original simple tuples.

Having that done and assuming that the system was capable of producing an output at the inference phase, that complied with the conventions of the *modified annotation* format, the next step was to work out a means of rearranging the arguments of the resulting *compound spatial tuples*, spawning several simple spatial tuples out of one *compound tuple*, determined by the system. In other words, the task was to perform *inverse transformation*. The first stage of the transformation was to separate the words assigned one of the *basic roles* (i.e., representing the *common part*) in the *compound tuple* from those assigned *surrogate roles*, creating two distinct sets. The next step was to reassign *basic roles* to the words, that the system had labeled with *surrogate roles* (i.e., to the words from the second set). Finally, it was necessary to create a bunch of tuples, one for each of the possible distinct combinations of the words from the second set and in further completing the resulting tuples with the words assigned common roles (i.e, from the first set). For example, as a result of the transformation, the tuple $next\_to(chair, bed, table, wardrobe)$, returned by NLPNET, would originate 4 distinct spatial relations: $next\_to(chair, table)$, $next\_to(bed, table)$, $next\_to(chair, wardrobe)$ and $next\_to(bed, wardrobe)$,

while each of the constituents of these tuples will be assigned a role label, that is accepted by the *original annotation* formalism.

We argue, that this transformation is consistent for all the examples present in the used dataset. Being applied, it produces a bunch of spatial tuples, that are in accordance with the original annotation format conventions. Moreover, all of these tuples can be qualified as *true positives* at the evaluation stage, given that there is a tuple in the corresponding *modified annotation* file, that is identical to the one determined by the system. Spatial relations convey information on the position of a potentially mobile entity (i.e., the *trajector*) with respect to the object that is treated as a reference (i.e., the *landmark*). On the one hand, the fact that there are several spatial relations sharing the same spatial pivot means, that the elements of the set of several (possibly one) potentially mobile objects are in the same position relatively to the members of the set of reference objects (this set may consist of only one landmark). On the other hand, no matter how many elements these two sets include, the position of every element of the first set is the same with respect to each of the elements of the second set. It is therefore equally legitimate to express the discussed spatial semantics using the formalism of simple spatial tuples or that of *compound* ones. The first was accepted by the original annotation format, while the last was proposed in this work and followed by the *modified annotation* files.

The limit of applicability of the discussed transformations is still an issue. The point is, that although the reasoning stated above can be used for a vast majority of cases, natural language still possesses a limited number of *modifiers* that change the semantic interpretation of a phrase containing multiple spatial relations that share the same spatial pivot. The word *respectively* is probably the most frequently used among such *modifiers*. When a phrase baring spatial semantic meaning contains a *modifier*, the corresponding *compound relation*, composed of a spatial pivot and $n$ spatial arguments belonging to each of the basic (i.e., non-surrogate) role classes, should be actually broken down into $n$ simple spatial relations instead of the greater number of relations that is spawned normally, i.e., when the modifiers are not observed in the sentence. Indeed, the *compound tuple* $next\_to(chair, bed, table, wardrobe)$, extracted at the inference stage from the phrase *There is a chair and a bed next to the table and the wardrobe respectively*, should originate the tuples $next\_to(chair, table)$ and $next\_to(bed, wardrobe)$ instead of the 4 tuples that had to be spawned in case the modifier *respectively* were absent.

A possible way to get round this difficulty may be to assign role types of additional *surrogate role classes* to landmarks that are part of *compound tuples* in the *modified annotation* file. These role classes would contain a reference to the trajector that is actually associated with the given landmark. In the example presented above, *table* should be labeled by the *1-landmark* tag in the *modified annotation* file, while *wardrobe* should receive the *2-landmark* label. These labels state that the first trajector (i.e., *chair*) is only associated with the first landmark (i.e., *table*), while the position of the second trajector (i.e., *bed*) is just relative to the second landmark (i.e., *wardrobe*) and not to all of them. After the system

manages to learn the new *surrogate roles* and is capable of tagging the examples at the inference stage correctly, it is possible to re-create the *simple* tuples out of a *compound* one, getting back to the role classes generally accepted by the SpRL formalism. We realize, however, that introduction of new role classes might produce a negative effect on the accuracy of the system-returned results. One way to facilitate the training process could be to encode the presence of a *modifier* in the feature vectors. The last possibility was not explored in this work and is one of the directions for future investigation.

The rest of the idea was outlined at the implementation phase. However, due to the absence of suitable examples in the training corpora (i.e., no phrases containing *modifiers* were found), we did not have the opportunity to test the robustness of the advanced version of the transformation as well as to discover its eventual limits of applicability.

### 3.4.2 Dynamic Relations

The model, that was obtained by training the system on the dynamic relations corpus without any modifications introduced to the input data format, showed considerably worse performance than the basic model for static spatial relations in terms of precision and recall. The reason was the greater number of tagging options (i.e., there are 7 possible spatial role classes for dynamic spatial relations instead of 3 for the static ones), the potentially bigger size of the tuples, the comparatively loose logical coupling between the elements of the long tuples, as well as the presence of various inconsistencies in the way the *gold annotation* examples were tagged. One more probable cause of inefficiency was that in the case of relations, containing both a motion and a spatial indicator, it was unclear which of the two had to be treated as the pivot of the relation and detected at the pivot identification stage. NLPNET does not permit that a relation contain more than one pivot, which led to the necessity of making the aforementioned choice. In either case, ignoring the pivotal role of a spatial or a motion indicator might lead to a decrease in the ability of the system to determine the spatial arguments of a pivot, that in reality was not being treated as such. The relations sharing the same pivot could also be one of the issues.

To improve the system's performance, certain alterations, discussed below, were introduced to the input data format. Having this done, a *modified annotation file*, that could be converted to the CoNLL style format and consequently fed to the system for training, was created. The output, obtained at the inference phase and supporting the rules of the *modified annotation* formalism, could be afterwards converted back to the original annotation format and treated as the system-produced output at the results evaluation phase.

The first of the alterations affected those tuples of the original annotation file, that contained both a spatial and a motion indicator or, in other words, that comprised both the *purely static* and the *purely dynamic* components. We call a tuple, formed by the spatial indicator, the landmark, the direction and the distance indicator of the original tuple the *purely static* component, while the rest of the constituents of

the original tuple are considered to form the *purely dynamic component*. The idea was to separate these two components, with the original spatial tuple resulting in two *daughter tuples* in the *modified annotation file*. The spatial and the motion indicator would serve as the pivots of the first and the second of the spawned tuples respectively. The expected system performance gain would be due to the possibility of determining both the spatial and the motion indicator at the pivot identification step. Moreover, it would allow for eliminating the cases of *paths* being erroneously detected as *landmarks* and vice versa, as the former would never occur in static relations, and the latter would result unlikely to be encountered as parts of dynamic relations. Finally, the constituents of the newly formed spatial tuples would arguably be more logically coupled, which could also produce a positive effect.

After being returned by the system at the inference phase, the tuples adhering to the formalism of the *modified annotation*, are passed through the *inverse transformation*. The pitfall of the approach being discussed lies in the fact that to reassemble the tuples during the *inverse transformation*, it is essential that the *purely dynamic* and the *purely static daughter tuples* contain a member in common, that would serve as the link between the two. It was decided that the word, serving as the motion indicator in the *purely dynamic daughter tuple*, would also be assigned a *trajector* label in the *purely static daughter tuple*. Hence, if the output of the system at the inference phase contains two tuples, that have an element in common, which is tagged as the motion indicator in one of them and as a trajector in the other, the *inverse transformation* would have them joined in one tuple with discarding the trajector label assigned to the word in common in the *purely static daughter tuple*.

To illustrate the said above, let us consider the phrase *I traveled some 150 kilometers in Alentejo on the way to Algarve*. In the original annotation file, this sentence would be represented by the tuple *traveled(I, some 150 kilometers, in, Alentejo, on the way to Algarve)*, where *I* is the trajector, *some 150 kilometers* is the distance indicator, *in* is the spatial indicator, *Alentejo* is the landmark, *on the way to Algarve* is the path, *traveled* is the motion indicator. The application of the transformation results in the following two *daughter tuples* in the *modified annotation* file: *traveled(I, on the way to Algarve)*, *in(traveled, some 150 kilometers, Alentejo)*. The first of the two *daughter relations* is *purely dynamic*, while the second is a *purely static* one. In the first tuple, *I* is the trajector, *traveled* is the motion indicator, serving as the pivot of the relation, *on the way to Algarve* is the path. In the second tuple, *in* is the spatial indicator, *Alentejo* is the landmark, *some 150 kilometers* is the distance indicator, while *traveled* is assigned a trajector label and will be used to link the two *daughter relations* to get the original tuple reassembled during the inverse transformation in the future. In case the system manages to determine the tuples *traveled(I, on the way to Algarve)* and *in(traveled, some 150 kilometers, Alentejo)* at the inference phase, the *inverse transformation* script will have the two assembled in one tuple (i.e., *traveled(I, some 150 kilometers, in, Alentejo, on the way to Algarve)*), while its constituents will be labeled according to the *original annotation* formalism.

Alongside the transformation, described above, the spatial relations of the *original annotation file* are subject to the transformation affecting the tuples that share the same spatial or motion pivot. It is quite analogous to the one applied for the case of the dataset, that consisted exclusively of static spatial relations.

We further exemplify the discussed transformation as a whole by the phrase *Even outside Europe and Asia, Australia is fun for us to travel from the North to warmer waves*. Sticking to the *original annotation* requirements, this sentence is represented by the following tuples: *outside(Australia, Europe)*, *outside(Australia, Asia)*, *travel(us, outside, Europe, from the North to warmer waves)* and *travel(us, outside, Asia, from the North to warmer waves)*, where *from the North to warmer waves* is the path in the third tuple, *Australia* is the trajector in the first and the second tuples, *Europe* is the landmark in the first and the third relations, *Asia* is the landmark in the second and the forth relations, *us* is the trajector in the third and the forth tuples, *outside* is the spatial indicator in all the relations where it is present, and *travel* is the motion indicator in the third and the forth tuples.

The first stage of the discussed transformation leads to the following set of tuples: *outside(Australia, Europe)*, *outside(Australia, Asia)*, *travel(us, from the North to warmer waves)*, *outside(travel, Europe)*, *outside(travel, Asia)*. All the elements of these tuples retain the labels assigned to them in the *original annotation*, except for the word *travel*, which is tagged with the trajector label in the last two relations.

On the next stage of the transformation, the spatial tuples, having the same pivot, are joined within one *compound relation*. As a result, the tuples *outside(Australia, travel, Europe, Asia)* and *travel(us, from the North to warmer waves)* will represent the analyzed phrase in the *modified annotation* file. In the first of the *daughter tuples*, *Australia* is tagged as *trajector-1*, *travel* is the *trajector-2*, *Europe* is the *landmark-1* and *Asia* is the *landmark-2*. The roles, assigned to the elements of the second *daughter tuple*, do not differ from the ones accepted by the *original annotation* formalism. The *modified annotation* file will be subsequently cast to the CoNLL format and used for system training.

If the system returns the tuples *outside(Australia, travel, Europe, Asia)* and *travel(us, from the North to warmer waves)* at the inference phase, the *inverse transformation* script should have them converted to the generally accepted annotation formalism. This task was solved in two steps. The first step is actually the *inverse transformation*, applied in the case of the dataset exclusively composed of static relations, that was described in the previous section. We get the following bunch of tuples: *outside(Australia, Europe)*, *outside(Australia, Asia)*, *travel(us, from the North to warmer waves)*, *outside(travel, Europe)*, *outside(travel, Asia)*. At the second step, the *purely dynamic tuple* (i.e., *travel(us, from the North to warmer waves)*) is joined to each of the tuples, whose trajector coincides with the motion indicator of the former (i.e., with the tuples *outside(travel, Europe)* and *outside(travel, Asia)*). Finally, we get the tuples *outside(Australia, Europe)*, *outside(Australia, Asia)*, *travel(us, outside, Europe, from the North to warmer waves)* and *travel(us, outside, Asia, from the North to warmer waves)*, which are in accordance with the

original annotation formalism.

The advanced version of the transformation, that concerned phrases containing *modifiers* (e.g., the word *respectively*), previously used for static relations and described in the previous section, is also applicable for dynamic relations. In this case, landmarks, direction and distance indicators, as well as paths in the phrases that contain *modifiers*, should have reference to the trajector they are associated to. New *surrogate role classes* should be introduced to be assigned to landmarks, directions, distances and paths, that are marked out in sentences where a *modifier* is observed.

No sentences, that could give us an opportunity to test the advanced version of the transformation were found in the *gold annotation* that was used. This task remains one of the lanes for future work.

## 3.5 Programming Technology

The scripts, aimed at improving the quality of the models, produced by NLPNET, as well as those for loading the word2vec word embeddings, needed to be compatible with the studied system. Taking into account that NLPNET was implemented in Python 2.7, we also used this programming language in our work. The advantages of the Python programming language are the intuitiveness of its syntax, the fact that its core and most of the packages are distributed with no restrictions and the well-engineered support of object-oriented programming (OOP) techniques. On the other hand, the language possesses certain flexibility and does not obey all the OOP rules very rigorously, which could complicate program development and may hardly be justifiable for small-scale or middle-scale systems.

Moreover, there are a lot of Python packages, that were conceived to support advanced mathematical operations and the data structures that are inherent to these operations. One of such packages, used in our work, is *NumPy*, which is a means for scientific computing with Python, that supports operations with N-dimension arrays, sophisticated mathematical functions and integral transforms, as well as contains various linear algebra capabilities.

Besides specialized packages for mathematics, the solution for the SpRL task relies on a number of packages, created especially for dealing with a wide range of NLP tasks. One important representative of such packages is *NLTK*, which provides functionality for word tokenizing (i.e., delimiting a sentence into words), dependency parsing and stemming. Another package, which was found extremely useful to load the *word2vec embeddings* and convert the respective files to a format accepted by NLPNET, was *Gensim*. Being a framework for vector space modeling, this package is a tool for dealing with natural language processing and information retrieval tasks, that require managing very large amounts of multidimensional data. This data may be represented by the training corpus or by the word embeddings, which was the case for our experiments. One interesting feature about Gensim is that the provided algorithms are *memory independent*, which means that the size of the processes data corpora can be

larger than the size of RAM on the computer.

The performance of the obtained models was measured with the help of the evaluation script, created for the purpose. It was written in the Python programming language as well.

## 3.6 Overview

In this chapter, we dwelt on the structure of the neural networks used for each of the sub-tasks that SpRL comprises (i.e., spatial pivot identification, argument identification and argument classification). We also provided some justifications for the parameter values used for the training process. The configurations of the neural networks we employed, as well as the concrete values of the training parameters, were strongly influenced by the experience, gained as a result of previous experiments with NLPNET. Although these experiments were dedicated to SRL rather than SpRL, it is possible to extrapolate most of the good practices for SRL with NLPNET to SpRL as well, thanks to the fundamental similarity between these two tasks.

We also gave coverage of the input data format used for system training, as well as of the transformations, that helped us to improve the quality of the models being obtained. One transformation was employed for static relations, and a combination of two distinct transformations was found useful for the case of dynamic relations.

Finally, we described the technology we used for the programming process and briefly mentioned the libraries we found especially useful to tackle the SpRL task at its different phases.

# 4

# Experimental Evaluation

## Contents

This chapter describes the experiments carried out in the course of this work. It covers in detail the datasets used for system training and evaluation, the leveraged annotation scheme and gives a survey of the employed evaluation metrics. It also characterizes various models for performing SpRL obtained as a result of the experiments with system training. The performance indicators for each model is finally presented and some conclusions are drawn.

## 4.1 Datasets and Annotation Format

In the experiments, we leveraged the corpus that was published in the context of the SemEval-2013 shared task. The dataset consists of two parts, with the first one containing annotations for static relations, and the other serving for describing static and dynamic spatial relations. Both of the corpora had been previously annotated according to the SemEval annotation scheme, with the MAE tool [28] being employed.

The first part is a subset of the IAPR TC-12 corpus [29], that contains annotations to photos taken by tourists. The phrases describe the objects that photos feature and the objects' position in relation to other objects in the image. The sub-corpus is a valuable source of spatial information, exposing most of the ways used by natural language to convey static spatial semantics. The sub-corpus consists of 1213 sentences. Following the recommendations of the organizers of the SemEval-2013 event, 613 sentences were selected for model evaluation, and the resting part of the corpus (i.e., 600 phrases) was used for system training. Note that in order to experiment with how NLPNET handled previously seen data at the inference phase, as well as to evaluate the effects of overfitting, we also induced models trained on the corpus, that would be further used for system evaluation.

The other part of the SemEval-2013 corpus was obtained from the Degree Confluence project. The content provides descriptions of sites at the intersections of certain latitudes and longitudes. It was produced by traveling enthusiasts and is a rich source of phrases bearing semantic meaning. The sub-corpus contains a total of 117 files with 1789 phrases. To be compliant with the SemEval-2013 evaluation methodology, an amount of 1422 sentences was used for model training, while the other 367 sentences were left for evaluation purposes. Analogously to the case of static relations, we also obtained models trained on the dataset, that was used for evaluation. Just like in the previous case, we aimed at estimating the influence of overfitting while dealing with previously seen data.

The annotations are placed in files, which can be processed as well-formed XML documents. Every XML tag represents one of the annotation elements (i.e., the analyzed text, the marked out spatial roles or the spatial tuples, composed of such spatial roles). Each of the supported spatial role classes (i.e., *TRAJECTOR*, *LANDMARK*, *SPATIAL_INDICATOR*, *MOTION_INDICATOR*, *PATH*, *DIRECTION* or *DISTANCE*) is represented by an XML tag with a distinct name, and the attributes are used to characterize

each of the spatial role class instances. Note that the annotation scheme is *span-based*, i.e., spatial meaning is assigned to chunks of tokens, rather than words, which would be typical for *head-based* annotation schemes. The role's attributes are the textual span that is attributed spatial meaning, the start and the end positions of the role span in the analyzed text, as well as the unique identifier of the spatial role instance. As for the XML elements, representing spatial tuples, they all have the same tag name (i.e., *RELATION*). The spatial roles, that the tuple comprises, are represented by attributes with respective names (i.e., *trajector_id*, *landmark_id*, *spatial_indicator_id*, etc.). The values of these attributes are the unique identifiers of the roles that form the tuple.

The following tables summarize the statistics related to the datasets, employed in the course of our work. They also point to how each of the sub-corpora was used at the SemEval-2013 shared task.

**Table 4.1:** Statistics for the Static Relations Corpora

| Used for | Sentences | Trajectors | Landmarks | Spatial Indicators | Relations |
|---|---|---|---|---|---|
| Training | 600 | 716 | 661 | 670 | 765 |
| Evaluation | 613 | 872 | 743 | 796 | 940 |

**Table 4.2:** Statistics for the Dynamic Relations Corpora

| Used for | Sentences | TR | LN | SPA | MOT | PAT | DIR | DIS | Relations |
|---|---|---|---|---|---|---|---|---|---|
| Training | 1422 | 1701 | 1037 | 879 | 1039 | 945 | 223 | 307 | 2105 |
| Evaluation | 367 | 497 | 316 | 247 | 305 | 240 | 37 | 87 | 598 |

Being a good example of phrases in *unrestricted* natural language, the sub-corpora used both for training and evaluation, contain a number of flaws, malformed sentences and errata. As a result, it sometimes appeared difficult to delimit the text into phrases, as well as to provide the most adequate spatial role annotations. The organizers of the SemEval-2013 challenge tried to rectify the data in order to make it more consistent and programmer-friendly [11]. Their another intention, however, was not to change considerably the original orthography and text formatting. Consequently, the quality of the models, obtained in the course of our work, might sometimes result or seem worse than it could have been if the corpus were annotated purely accurately.

## 4.2 Models under Evaluation

The parameters used for the model training process, as well as the neural network configurations employed, are described in sections 3.1 and 3.2 of chapter 3.

The performance of four distinct models was evaluated for the static relations dataset. The *first model* was obtained by training the NLPNET system on the *gold annotation* file containing 600 sentences and was evaluated using the corpus of 613 phrases. Note that the training and the evaluation sub-corpora

had no sentences in common, i.e, the model was evaluated on unknown data. For this model, we did not use any input data format transformations, relying exclusively on pre-trained *word2vec embeddings* and the capacity of the system to deal with the SRL task, given the affinity between SRL and SpRL. The *second model* was trained and tested just like the first model, but it also used the proposed input data format transformations to improve the model quality. The *third model* was obtained by training the system on the corpus of 613 sentences, that was further used for evaluation. Therefore, this time we evaluated system performance on previously seen data. Analogously to the first model, this model used pre-trained *word2vec embeddings* with no additional input data format transformations. The fourth model was trained and tested in conditions, identical to those for the third model. However, it also leveraged the input data format transformations for the training phase to achieve a gain in the evaluation metrics.

Analogously to the case of static relations, four distinct models were obtained in our experiments with the dynamic relations dataset. The first two models were trained on a subset of the Confluence Degree project containing 1422 sentences and were further evaluated on the sub-corpus of 367 phrases, i.e., on unknown data. The *first model* relied on *word2vec embeddings* as the main features and did not leverage any additional input data format transformations. The *second model* used both the word embeddings and the proposed input data format transformations for the training phase in the strive to boost the values of the performance metrics. The *third* and the *fourth* models were induced on the sub-corpus of 367 phrases and were evaluated using the same dataset as the gold annotation, i.e., on previously seen data. Unlike the third model, the fourth model made use of the proposed input data format transformations.

## 4.3 Evaluation Methodology

The evaluation methodology is influenced by the one used at the previous SemEval challenges, where the SpRL task was discussed. The SpRL process was divided into two sub-tasks, that were subject to separate evaluation. The first sub-task centers on determining individual spatial roles, while the second one focuses on evaluating the model's capability to determine complete spatial relations.

More specifically, the first sub-task was to determine chunks of tokens, assigned a certain spatial role label. The spatial roles are used as annotations, and a spatial role is considered correctly determined if there is an overlap of at least one character with what is seen in the *gold annotation*, and also if the role class coincides with the role class in the *gold annotation*. The described evaluation criterion is often referred to as the *relaxed* one, as opposed to the *strict* criterion, which compares the system output with the *gold annotation* file and expects not only an exact match in the role classes, but also a precise superposition between the system-determined role span and the text of the corresponding role in the *gold annotation*. Note, that using the *strict* criterion for the sub-task was also considered by the

organizers of the previous SemEval events. Therefore, we do not discard the possibility of leveraging it in our eventual future experiments as well.

It is important to point out, that to be compliant with the evaluation methodology, used by the participants of the previous SemEval events, we postulate that a spatial role may be determined only once, no matter of how many spatial relations it appears to be part of. This assertion was natural for evaluating the previously presented systems for SpRL, because practically all of them viewed the process as a two-step procedure. The first step was dedicated to determining the spatial roles, while the second one consisted in assembling the previously rendered spatial roles in spatial tuples. SpRL with NLPNET is based on a slightly different philosophy, with the spatial role spans being marked out for a given previously determined spatial pivot. That is why, the determining of spatial roles and the tuples assembling might be arguably seen as inherent to a single process. In other words, each of the spatial roles, rendered by NLPNET, was actually determined as many times, as the number of system-defined tuples, that it appears to be part of. However, the SemEval-style evaluation methodology has been more customary for SpRL and is supported by its founding fathers. We stick to it as well, although while analyzing the results we shall take into account that the main destination of NLPNET is to determine spatial role tuples, rather than individual spatial roles. Hence, we would value the improvements in the results for the second sub-task (i.e., complete spatial role labeling) and consider this eventual improvement a bigger success.

The second sub-task focuses on evaluating the system's ability to identify the spatial relations, i.e., arrange the determined spatial roles into spatial tuples. The *strict* evalution criteria is used for the sub-task. Each tuple is considered correct if it consists of the same spatial roles, as the referenced tuple in the *gold annotation*, while every spatial role in the system-assembled tuple matches each role in the *gold annotation* tuple. A match for spatial roles is understood here in terms of the criterion employed for evaluating the system performance for the first sub-task (i.e., in our case we follow the *relaxed* criterion requirements).

## 4.4 Evaluation Metrics

Following the established practice, the *precision*, *recall* and the *F1* indicators were employed to measure how good the system's performance was regarding each of the sub-tasks subject to separate evaluation. All these metrics build on the notions of true positive, false positive and false negative results.

A result is considered a *true positive* if the corresponding instance was identified correctly according to the stipulated evaluation criteria. A *false positive*, also known as an *error of the first kind*, is a system-retrieved instance, which can not be considered true positive according to the evaluation criteria used. *False negatives* (also known as *errors of the second kind*) are instances, that possess all the character-

istics that the legitimate instances have, and as such, should have been retrieved by the model. These instances, however, were not determined due to the tested model not being perfect.

In the course of our experiments, we considered *true positive* those system-determined spatial roles or tuples, for which matches, suitable according to the used evaluation criteria, were found in the *gold annotation*. *False positive* spatial roles or tuples were the system-determined instances, for which no corresponding annotation, adequate with respect to the employed evaluation criteria, could be indicated in the dataset used for evaluation. Finally, false negatives are the elements of the *gold annotation* that were not determined by the model at the inference phase.

Upon gathering the statistics of the number of true positives $tp$, false positives $fp$ and false negatives $fn$, we can determine precision $P$ and recall $R$, using the following formulas.

$$P = \frac{tp}{tp + fp} \tag{4.1}$$

$$R = \frac{tp}{tp + fn} \tag{4.2}$$

*Precision*, also called *positive predictive value*, measures the quality of the results rendered by the system. It can also be seen as a characteristic of how useful the obtained results are. The denominator of the formula for precision contains the total number of instances that were identified by the system, whose performance is being evaluated. Getting a precision of 1.0 would mean the absence of *errors of the first kind*, i.e., that all the retrieved results are true positives. In our case, it would signalize that all the spatial roles or tuples, that were marked out by the SpRL process, are actually correct with respect to the employed evaluation criteria.

The *recall* metric, sometimes also referred to as *sensitivity*, characterizes the completeness of the obtained results. The denominator of the recall formula is the total number of the instances that are actually legitimate, i.e., the number of spatial tuples or elements of the given spatial role class in the gold annotation. In the ideal case, the recall value of 1.0 would mean the absence of *errors of the second kind*. In our case, it would mark that all the spatial role instances, that the model was supposed to identify, were actually rendered, although it is still possible that some incorrect instances (i.e., *false positives*) were erroneously included in the results set.

Precision is often increased at the cost of decreasing recall and vice versa. Consequently, a metric should be introduced that would represent a sort of a mean value of the two. This metric should also take into consideration, whether precision and recall are equally important or if a high value of one of the metrics is more preferable than a high value of the other. In fact, this decision strongly depends on the task specifics. In our experiments, we did not see any reason to prioritize one of the metrics rather than the other, hence we consider the two metrics to be of equal importance.

The $F1$ score is the harmonic mean of precision and recall and is typically used whenever there is

no need to favor a high value of one of the metrics over the other. It can be formally defined as follows:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{4.3}$$

## 4.5 Experimental Results

This section presents the experimental results, obtained for each of the models, described in section 4.2.

### 4.5.1 Static Relations

We further present the experimental evaluation results for the four distinct models, obtained on the IAPR-12 dataset containing only static spatial relations.

#### 4.5.1.A The First Model

The model was trained on the subset of the IAPR-12 corpus, that contained 600 sentences, and was evaluated based on another subset of 613 phrases.

**Table 4.3:** Static Relations. Performance Evaluation Results for the First Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.772 | 0.350 | 0.482 |
| LANDMARK | 0.903 | 0.688 | 0.781 |
| SPATIAL_INDICATOR | 0.957 | 0.753 | 0.843 |
| COMPOSITE (ROLES) | 0.891 | 0.579 | 0.702 |

**Table 4.4:** Static Relations. Performance Evaluation Results for the First Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.379 | 0.282 | 0.323 |

Looking at the obtained results, we can state that the model is good at recognizing individual spatial roles (with the exception of trajectors) and is characterized by moderate performance as far as spatial tuple recognition is concerned. The low value of recall for trajectors and complete spatial tuples identification is due to the model's inability to determine the arguments for spatial indicators that serve as pivots of various relations. The model can render only one of the bunch of relations, that share the same spatial pivot. The roles that are part of other relations are neglected, which results in a considerable number of false negatives. The survey of the *gold annotation*, used for evaluation, revealed that the majority of spatial relations, that had a spatial pivot in common, also shared the landmark. This fact explains why

there are much more false negatives among trajectors than among landmarks and, therefore, the reason for the recall value for trajectors being much lower than for landmarks.

#### 4.5.1.B The Second Model

The model was trained on the subset of the IAPR-12 corpus, that contained 600 sentences, and was evaluated based on another subset of 613 phrases. It also leveraged the proposed input data format transformations.

**Table 4.5:** Static Relations. Performance Evaluation Results for the Second Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.761 | 0.556 | 0.643 |
| LANDMARK | 0.866 | 0.693 | 0.770 |
| SPATIAL_INDICATOR | 0.945 | 0.755 | 0.840 |
| COMPOSITE (ROLES) | 0.857 | 0.662 | 0.747 |

**Table 4.6:** Static Relations. Performance Evaluation Results for the Second Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.500 | 0.425 | 0.460 |

We can observe that the second model proved to be more capable of recognizing complete spatial tuples than the first model, with both the results for precision and recall having improved. We can also see that the values of recall have improved for all the role classes in comparison with the corresponding results obtained for the first model. The values of precision have slightly deteriorated for all the role classes. However, the overall negative effect of it did not affect the F1 metric for trajectors and for roles of all classes indiscriminately (i.e., the value of F1 has improved for these cases).

#### 4.5.1.C The Third Model

The model was trained on the subset of the IAPR-12 corpus, that contained 613 sentences, and evaluated using the same dataset as the gold annotation.

**Table 4.7:** Static Relations. Performance Evaluation Results for the Third Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.965 | 0.701 | 0.812 |
| LANDMARK | 0.961 | 0.818 | 0.884 |
| SPATIAL_INDICATOR | 0.958 | 0.839 | 0.895 |
| COMPOSITE (ROLES) | 0.961 | 0.780 | 0.861 |

**Table 4.8:** Static Relations. Performance Evaluation Results for the Third Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.774 | 0.628 | 0.693 |

Comparing the results for the third model with those obtained for the first model, we can see evidences of *overfitting*, although it is not that clearly notable for all the role classes with the exception of trajectors. Similarly to the first model, the bottleneck of the model's performance is the recall value, obtained for the task of recognizing trajectors.

#### 4.5.1.D The Fourth Model

The model was trained on the subset of the IAPR-12 corpus, that contained 613 sentences, and further on was evaluated using the same dataset as the gold annotation. It also leveraged the proposed input data format transformations.

**Table 4.9:** Static Relations. Performance Evaluation Results for the Fourth Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.982 | 0.954 | 0.968 |
| LANDMARK | 0.967 | 0.926 | 0.946 |
| SPATIAL_INDICATOR | 0.961 | 0.864 | 0.910 |
| COMPOSITE (ROLES) | 0.971 | 0.915 | 0.942 |

**Table 4.10:** Static Relations. Performance Evaluation Results for the Fourth Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.905 | 0.885 | 0.895 |

Comparing the results for the fourth model with those for the second one, we can see evidences of *overfitting*, which are more salient for the more complex sub-task, i.e., for complete spatial relation identification. Comparing the performance of the fourth model with that of the third one, we can observe a steady boost in all the evaluation metrics.

### 4.5.2 Dynamic Relations

We further present the experimental evaluation results for the four distinct models, obtained on the Confluence Degree project dataset containing both static and dynamic spatial relations.

#### 4.5.2.A The First Model

The model was trained on the subset of the Confluence Degree corpus, that contained 1422 sentences, and was evaluated based on another subset of 367 phrases.

**Table 4.11:** Dynamic Relations. Performance Evaluation Results for the First Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.755 | 0.174 | 0.283 |
| LANDMARK | 0.575 | 0.077 | 0.136 |
| SPATIAL_INDICATOR | 0.446 | 0.101 | 0.165 |
| MOTION_INDICATOR | 0.849 | 0.318 | 0.463 |
| PATH | 0.866 | 0.242 | 0.378 |
| DIRECTION | 0.500 | 0.024 | 0.051 |
| DISTANCE | 1.000 | 0.080 | 0.148 |
| COMPOSITE (ROLES) | 0.749 | 0.174 | 0.282 |

**Table 4.12:** Dynamic Relations. Performance Evaluation Results for the First Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.317 | 0.087 | 0.137 |

Judging by the F1 score, we can state that the model is most capable of detecting motion indicators and paths. In fact, it is more efficient at recognizing the former, but more precise while detecting the latter. The first proves that it is relatively easy to mark out motion indicators, that serve as pivots of dynamic relations. Interestingly, the model does not mark out many false positives among paths, which could be explained by the fact that most of the sentences in the evaluation corpus do contain paths, and by that the position of paths in a phrase is usually fixed, i.e., a path usually comes immediately after a motion indicator. The model returns rather precise results for direction and distance indicators, but the capability of recognizing these two spatial roles (i.e., the recall) is low. This can be justified by the relatively few number of examples of sentences containing directions and distance indicators in the training dataset. According to the requirements of the training input data format used by NLPNET, a spatial indicator, that is part of dynamic relations (i.e., tuples comprising a motion indicator), cannot be treated as the second pivot of the relation. This fact probably explains why the performance indicators for spatial pivots are not that high. The model is characterized by moderate performance in terms of detecting trajectors and landmarks, which may be explained by their non-fixed positions relative to a motion indicator and the semantic similarity between a landmark and a path or a direction indicator (i.e., some of the landmarks were tagged as directions or paths).

The model did not prove to deal with complete spatial tuple identification efficiently, which is explained by the complexity of dynamic spatial semantics and by the strict evaluation criteria employed for testing its capabilities concerning the second sub-task.

#### 4.5.2.B The Second Model

The model was trained on the subset of the Confluence Degree corpus, that contained 1422 sentences, and was evaluated based on another subset of 367 phrases. It also leveraged the proposed

input data format transformations.

**Table 4.13:** Dynamic Relations. Performance Evaluation Results for the Second Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.753 | 0.265 | 0.392 |
| LANDMARK | 0.639 | 0.153 | 0.247 |
| SPATIAL_INDICATOR | 0.550 | 0.223 | 0.317 |
| MOTION_INDICATOR | 0.808 | 0.331 | 0.470 |
| PATH | 0.732 | 0.250 | 0.373 |
| DIRECTION | 0.800 | 0.108 | 0.190 |
| DISTANCE | 1.000 | 0.207 | 0.343 |
| COMPOSITE (ROLES) | 0.719 | 0.242 | 0.362 |

**Table 4.14:** Dynamic Relations. Performance Evaluation Results for the Second Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.250 | 0.108 | 0.151 |

We can observe that the second model proved to be more capable of recognizing complete spatial tuples than the first model, with both the results for precision and recall having improved.

We can also see that the values of recall have improved for all the role classes in comparison with the corresponding results obtained for the first model. The values of precision have deteriorated for trajectors, paths and motion indicators. For trajectors and motion indicators, this did not result in a decrease in the F1 value. For paths, however, we still see a slight decrease in the F1 value, which can be explained by the fact that the tagging of these roles is strongly dependent on the pivot identification phase. The precision for motion indicators (i.e., for pivot identification) experiences a decrease, due to the appearance of two classes of pivots (i.e., spatial and motion indicators) after the input data format transformation was applied. This fact results in a bigger number of false positives.

The model is better than the first one at identifying spatial indicators and landmarks both concerning precision and recall.

It is also better than the first model in terms of the F1 score for spatial roles of all classes indiscriminately.

#### 4.5.2.C The Third Model

The model was trained on the subset of the Confluence Degree corpus, that contained 367 sentences and was evaluated based on the same dataset. It leveraged no input data format transformations.

**Table 4.15:** Dynamic Relations. Performance Evaluation Results for the Third Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.879 | 0.236 | 0.372 |
| LANDMARK | 0.895 | 0.168 | 0.283 |
| SPATIAL_INDICATOR | 0.788 | 0.166 | 0.274 |
| MOTION_INDICATOR | 0.928 | 0.338 | 0.496 |
| PATH | 0.943 | 0.346 | 0.506 |
| DIRECTION | 0.800 | 0.216 | 0.340 |
| DISTANCE | 1.000 | 0.230 | 0.374 |
| COMPOSITE (ROLES) | 0.899 | 0.249 | 0.390 |

**Table 4.16:** Dynamic Relations. Performance Evaluation Results for the Third Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.610 | 0.186 | 0.285 |

The results obtained for the third model are rather modest, taking into account the fact that it was evaluated on previously seen data. This is justified by the small size of the corpus used for system training. Comparing the results for the third model with those obtained for the first model, we can see evidences of *overfitting*. Effectively, all the scores for all the items are higher for the third model than the corresponding results for the first model, although the effects of overfitting are not that sharp. Similarly to the first model, the model is good at identifying motion indicators and paths and is rather precise but not complete while tagging direction and distance indicators. Just like for the first model, the bottlenecks are the recall values obtained for landmarks and spatial indicators.

#### 4.5.2.D The Fourth Model

The model was trained on the subset of the Confluence Degree corpus, that contained 367 sentences and was evaluated based on the same dataset. It leveraged the proposed input data format transformations.

**Table 4.17:** Dynamic Relations. Performance Evaluation Results for the Fourth Model (the First Sub-task)

| Spatial Role Class | Precision | Recall | F1 |
|---|---|---|---|
| TRAJECTOR | 0.843 | 0.241 | 0.375 |
| LANDMARK | 0.825 | 0.173 | 0.286 |
| SPATIAL_INDICATOR | 0.772 | 0.203 | 0.321 |
| MOTION_INDICATOR | 0.917 | 0.340 | 0.496 |
| PATH | 0.943 | 0.342 | 0.502 |
| DIRECTION | 0.600 | 0.270 | 0.372 |
| DISTANCE | 1.000 | 0.310 | 0.473 |
| COMPOSITE (ROLES) | 0.869 | 0.259 | 0.399 |

**Table 4.18:** Dynamic Relations. Performance Evaluation Results for the Fourth Model (the Second Sub-task)

| Class Label | Precision | Recall | F1 |
|---|---|---|---|
| RELATION | 0.624 | 0.201 | 0.304 |

Similarly to the third model, the results are relatively moderate, considering the fact that the model was evaluated on previously seen data. This is explained by the small size of the training corpus. Comparing the results for the fourth model with those for the second one, we can see evidences of *overfitting*, which are more salient for the more complex sub-task, i.e., for complete spatial relation identification. Comparing the performance of the fourth model with that of the third one, we can observe trends, similar to those we mentioned while discussing the second model in comparison with the first one. They are an increase in the model's capability to recognize spatial indicators, trajectors and landmarks and a slight decrease in the performance indicator for paths. The model's capability to perform complete spatial relation identification, as well as the indicators for detecting spatial roles of all classes indiscriminately, are better with respect to the third model.

## 4.6 Overview

This chapter was dedicated to the results of implementing our ideas in practice. We started by describing the datasets used for system performance evaluation, presented some statistical data about the referred corpora and covered in detail the annotation format employed. The chapter also characterized the models, whose performance was evaluated and pointed to the differences and similarities between the conditions in which these models had been obtained. By leveraging four distinct models we did not only aim at putting to the test the effects of the proposed input data format transformations, but also pretended to evaluate how the system processed well-known data.

Further on, we dwelt on the used evaluation methodology, focusing on the features, that are peculiar for a span-based evaluation scheme. A two-step evaluation methodology was proposed that was compliant with the methodology for evaluating previously created systems for SpRL. We further gave attention to the precision, recall and F1 metrics, on which quantitative evaluation was based.

Finally, we presented the results obtained for each of the models and described the main trends observed. For individual spatial role identification, the leveraged input data format transformation resulted in increased values of recall and slightly deteriorated precision. For complete spatial tuple identification, the proposed transformations provided for obtaining better models not only in terms of recall, but also in terms of precision. The values of the F1 metric, obtained both for individual role identification and for complete tuple recognition, have improved for the models leveraging the proposed input data format transformation. The effects of overfitting are more clearly observed for more complex tasks, namely for complete spatial tuple identification rather than individual role determination.

# 5

# Conclusion

## Contents

This chapter concludes the MSc thesis and discusses the results achieved from a critical perspective. It points at the system's limitations and also mentions some interesting topics that could be the subject of further investigation in the field.

## 5.1 Conclusions

This work was dedicated to the SpRL task that emerged as a variation of SRL. Our idea was to choose the most efficient approaches for SRL and apply them to SpRL. The analysis of the techniques, previously proposed for SRL, and the systems, designed leveraging these approaches, drew our attention towards solutions, that made use of multilayer convolutional neural networks. One remarkable representative here is SENNA, which was a pioneer among systems of this kind. The NLPNET system, which is a Python re-implementation of SENNA, was designed specifically for dealing with SRL and POS tagging. This system was deemed especially useful for being adopted for SpRL.

Our approach contrasted with the approaches that had been previously leveraged to create the existing systems for SpRL [7, 8, 30]. The goal we pursued was to move away from using explicit linguistic knowledge, different heuristics and linguistically motivated feature engineering. In fact, our solution did not rely on any of the aforementioned mechanisms, but managed to achieve performance that is comparable to what had been reported by the authors of the known state-of-the-art solutions for SpRL.

Instead of leveraging linguistically motivated feature engineering and heuristics, our approach relied on word embeddings as the main features. By means of word embeddings, it is possible to capture the semantic relations between natural language concepts, with those relations sometimes being rather fine. One notable advantage of word embeddings is the fact that they are trained in an unsupervised fashion. Furthermore, the convolutional network provided by NLPNET, is capable of inducing new relevant features. We can state, that making use of the combination of word embeddings and the convolutional neural network approach is in line with the tendencies, that are nowadays widely accepted by the machine learning community. These tendencies aim at reducing the size of the corpus used for system training, gradually moving from supervised machine learning solutions towards self-learning systems.

Effectively, the basic benchmark versions of NLPNET models treated SpRL as SRL with an unconventional set of tags. These models were characterized by generally tolerable performance in terms of individual role recognition. This is true both for static and dynamic relations, although the results obtained for the dynamic case were worse, which is explained by greater complexity of dynamic spatial semantics and some inconsistencies in the training corpus. The recall values proved to be the weak point both for static and dynamic individual spatial role labeling. The performance for complete spatial tuple identification, especially for dynamic spatial relations, was quite modest, which can be explained by the latter task being much more complex than individual role recognition.

The proposed input data format transformations aimed at adjusting the existing annotation scheme, which had been originally designed for the participants of the previous SemEval events, to be most effectively interpreted by NLPNET at the model training stage.

For static relations, the transformation provided for a general increase in the values of recall for individual role labeling. For complete tuple recognition, the F1 metric experienced an increase of 42% for unknown data and of 30% for previously seen data. Although we had to sacrifice precision for some of the role classes, the F1 metric for individual role determination regardless of the class raised by 6 % for unknown data and by 9% for previously seen data.

For dynamic relations, the transformation provided for an increase by 10% in the F1 result for individual spatial roles in the case of previously unknown data and for a rise by 2.3% in the F1 result for the case of previously seen data. For complete tuple determination, the increase was of 28% for previously unseen data and of 6.7% in the case of previously seen data.

We believe, that the obtained results prove the legitimacy of tackling the SpRL task using the proposed methodology. The performance of our solution is slightly worse than the indicators, shown by the known state-of-the-art systems for SpRL [7,8]. However, they are a bit better than those for the UNITOR system [30], that was evaluated at the SemEval-2013 shared task. Although striving to improve the performance indicators was one of our priorities, we tried to refrain from using auxiliary linguistic knowledge and the output of other NLP tasks whenever possible. Being guided by this principle, we discarded the use of POS tags as components of the feature vectors, which could have most likely produced a positive effect on the evaluation metrics because of the correlation between the part of speech tag of a word and the role that it can play in a sentence (i.e., trajectors are usually nouns, landmarks are either nouns or adverbs, spatial indicators are mostly prepositions etc). Due to the relatively modest computation power at our disposal, we also had to limit the number of rounds for model training, seeking a compromise between the quality of the obtained models and the time spent on getting them. Finally, there are some interesting ideas we did not implement because of time limitations, that should undoubtedly result in a performance gain in our eventual future experiments.

## 5.2 System Limitations

Being based on the convolutional neural network approach and striving to fully exploit its benefits, the system possesses a number of weak points, which are inherent to this approach and to machine learning techniques in general.

NLPNET relies on word embeddings, trained in an unsupervised fashion, and the employed network architecture provides for effective capturing of new relevant features. However, it is still a solution based on supervised machine learning. The insufficient size of the annotated training corpus has always been

a problem for supervised machine learning systems. A common indicator of scarcity of training data is overfitting, of which we saw evidences in our experiments. In NLP tasks, centered on semantics, this difficulty may also be interpreted as the inability of the model to learn many of the ways that a natural language uses to convey certain meaning. The problem of scarcity of training data, however, was mitigated by using the large corpus of word2vec embeddings, which contains information on semantic relations between different natural language concepts and is equally applicable for many other related NLP tasks.

The implication of making use of a large data corpus is the need for considerable computational resources. The file containing word2vec embeddings weights around 7Gb, and special programming techniques should be employed to load it into the Random Access Memory (RAM) of an ordinary personal computer. Moreover, the complex topology of the convolutional network used by NLPNET is responsible for considerable time costs for model training and inference. In fact, thanks to the flexibility of the leveraged approach, the reported quantitative results could have been better should we have possessed more computational resources.

The proposed format transformations applied to the annotated data, fed to the system as input at the training phase, provide for representing a vast majority of cases typical for spatial role semantics as opposed to the one accepted in traditional semantic role labeling. Still, we have to admit that the transformations do not cover all the possible interpretations that phrases with spatial semantics could get. We did not have a chance to test our solution on a number of rather rarely used examples, such as the phrase *There is a chair and a bed next to the table and the wardrobe respectively*, which actually contains 2 spatial relations instead of 4. Note, that this irregularity is due to the presence of the word *respectively* in the phrase. Unfortunately, it is not yet absolutely clear, whether this case may be treated without using explicit information on the presence of *modifiers* like *respectively* in the analyzed sentence and encoding it in word embeddings.

## 5.3 Future Work

We could consider a number of directions for our eventual future investigations concerning the SpRL task. First, there is no doubt that the obtained results are far from being satisfactory considering the possibility of practical application of the achievements of SpRL in the neighboring fields. The solution for SpRL might still use some linguistically motivated features, such as POS tags, to ensure better results, although this is not fully in line with avoiding using explicit linguistic knowledge.

Classifying the obtained spatial relations into regional, directional and distal is one more task that we did not address due to time limitations, but that should be given attention in the future.

In our eventual future research, we should also consider making use of the method, recently ex-

pressed by P. Kordjamshidi et al [31]. The main idea of the approach is to leverage *Visually Informed Embeddings of Words (VIEW)*, induced on a large corpus of annotated photos. The obtained embeddings would store information on spatial arrangements between objects, that are common to real life spatial semantics. The VIEWs may be concatenated with the used word2vec representations for words, that are further loaded by the NLPNET system at the training stage. We believe, that this approach is a good alternative to using heuristics in order to prevent the system from deducing tuples that contradict to common sense spatial semantics. We could thus tackle the problem of the probable extracting of the illegitimate tuple *above(wall, bed)* from the phrase *There is a picture on the wall above the bed*, raised in [8].

In a more remote perspective, we can also experiment with the high-order low-rank tensor approach, that only recently emerged, but managed to achieve impressive results for SRL. Given the affinity between SRL and SpRL, this approach is promising for addressing the latter task as well.

# Bibliography


[1] R. Collobert, J. Weston, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (Almost) from Scratch,” *Journal of Machine Learning Research*, vol. 12, 2011.

[2] N. Kwon, M. Fleischman, and E. Hovy, “Senseval Automatic Labeling of Semantic Roles Using Maximum Entropy Models,” *Proceedings of the International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 2004.

[3] X. Carreras and L. Marquez, “Introduction to the CoNLL-2004 Shared Task. Semantic Role Labeling,” *Proceedings of the Conference on Natural Language Learning*, 2004.

[4] ——, “Introduction to the CoNLL-2005 shared Task. Semantic Role Labeling,” *Proceedings of the Conference on Natural Language Learning*, 2005.

[5] Y. Liu and A. Sarkar, “Experimental Evaluation of LTAG-based Features for Semantic Role Labeling,” *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2007.

[6] P. Kordjamshidi, M. V. Otterlo, and M. F. Moens, “Spatial Role Labeling. Task Description,” *Proceedings of the International Workshop on Semantic Evaluation*, 2012.

[7] P. Kordjamshidi, M. V. Otterlo, and M.-F. Moens, “Spatial Role Labeling: Towards Extraction of Spatial Relations from Natural Language,” *ACM Transactions on Speech and Language Processing*, vol. 8, no. 3, 2011.

[8] K. Roberts and S. M. Harabagiu, “UTD-SpRL: A Joint Approach to SpRL,” *Proceedings of the International Workshop on Semantic Evaluation*, vol. 2, 2012.

[9] E. R. Fonseca and J. L. G. Rosa, “A Two-step Convolutional Neural Network Approach for Semantic Role Labeling,” *Proceedings of the International Joint Conference on Neural Networks*, 2013.

[10] P. Kordjamshidi, M. V. Otterlo, and M. F. Moens, “Spatial Role Labeling: Task Definition and Annotation Scheme,” *Proceedings of the International Conference on Language Resources and Evaluation*, 2010.

[11] P. Kordjamshidi, O. Kolomiyets, S. Bethard, and M.-F. Moens, “Spatial Role Labeling. Task Description,” *Proceedings of the International Workshop on Semantic Evaluation*, 2013.

[12] J. Pustejovsky, P. Kordjamshidi, M.-F. Moens, and Z. Yocum, “SemEval-2015 Task 8. SpaceEval,” *Proceedings of the International Workshop on Semantic Evaluation*, 2015.

[13] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] S. Russel and P. Norwig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[15] M. Collins, “Tagging with Hidden Markov Models,” *Course notes for NLP by Michael Collins, Columbia University*, 2011.

[16] C. Sutton and A. McCallum, “An Introduction to Conditional Candom Fields,” *Foundations and Trends in Machine Learning*, vol. 4, no. 4, 2012.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” *CoRR*, vol. abs/1301.3781, 2013.

[18] K. Lund and C. Burgess, “Producing High-Dimensional Semantic Spaces from Lexical Co-occurrence,” *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, 1996.

[19] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global Vectors for Word Representations,” *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[20] T. Lei, Y. Zhang, L. Màrquez, A. Moschitti, and R. Barzilay, “High-order Low-rank Tensors for Spatial Role Labeling,” *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, vol. 15, 2015.

[21] T. Lei, Y. Zhang, T. Jaakkola, and R. Barzilay, “Low-Rank Tensors for Scoring Dependency Structures,” *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.

[22] J. Hajič, M. Ciaramita, and R. Johansson, “CoNLL-2009 Shared Task Description,” *Proceedings of the Conference on Natural Language Learning*, 2009.

[23] M. Roth and K. Woodsen, “Composition of Word Representations Improves Semantic role Labelling,” *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[24] P. Kordjamshidi, S. Bethard, and M.-F. Moens, “SemEval-2012 Task 3: Spatial Role Labeling,” *Proceedings of the International Workshop on Semantic Evaluation*, 2012.

[25] K. Litkovski and O. Hargraves, “SemEval-2007 Task 06: Word Sense Disambiguation of Prepositions,” *Proceedings of the International Workshop on Semantic Evaluation*, 2007.

[26] E. Agirre, L. Màrquez, and R. Wicentowski, “SemEval-2007: 4th International Workshop on Semantic Evaluation,” *Proceedings of the International Workshop on Semantic Evaluation*, 2007.

[27] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “Liblinear: A Library for Large Linear Classification,” *Journal of Machine Learning Research*, vol. 9, 2008.

[28] A. Stubbs, “MAE and MAI: Lightweight Annotation and Adjudication Tools,” *Proceedings of the 5th Linguistic Annotation Workshop*, 2011.

[29] H. J. Escalante, C. A. Hernández, J. A. Gonzalez, A. López-López, M. Montes, E. F. Morales, L. Enrique Sucar, L. Villaseñor, and M. Grubinger, “The Segmented and Annotated IAPR TC-12 Benchmark,” *Computer Vision and Image Understanding*, no. 4, 2010.

[30] E. Bastianelli and D. N. Danilo Croce, Roberto Basili, “UNITOR-HMM-TK: Structured Kernel-based Learning for Spatial Role Labeling,” *Proceedings of the International Workshop on Semantic Evaluation*, 2013.

[31] P. Kordjamshidi, O. Ludwig, X. Liu, and M.-F. Moens, “Deep Embedding for Spatial Role Labeling,” *arXiv preprint arXiv:1603.08474*, 2016.