Understanding Spatial Semantics in Natural Language

Alexey Mazalov, *Student, IST*

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.

Index Terms—Machine Learning, Convolutional Neural Networks, Natural Language Semantics, Spatial Role Labeling, NLPNET.

1 Introduction

Research in the area of Natural Language Processing (NLP) has started to deal with computational semantic analysis, tackling the issue of attributing meaning to a piece of text.

The Spatial Role Labeling (SpRL) task can be defined as the task of assigning several general spatial roles (e.g., *trajector*, *landmark*, *distance*, or *direction*) to the words/phrases in a sentence.

Semantic Role Labeling (SRL) is a task that is very similar to SpRL and from which the latter actually originated. The aim of SRL is to detect, predict and attribute a semantic role (e.g., *agent*, *predicate*, *beneciary*, etc.) to a word/phrase in a sentence.

1.1 Thesis Proposal

In this MSc thesis, we intend to adapt the existing solutions for the SRL task, which leverage modern machine learning techniques such as convolutional neural networks, for performing SpRL.

We consider our work a step away from feature engineering and using explicit linguistic knowledge for SpRL. The NLPNET 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.

Besides relying on a convolutional neural network, our approach also builds on using *word embeddings*, trained in an unsupervised fashion, as the main features.

1.2 Contributions

- The NLPNET system, originally designed to tackle the SRL task, has been reconfigured to deal with the SpRL task.
- The usefulness of leveraging word embeddings, trained in an unsupervised fashion, was demonstrated.
- Different neural network configuration modifications were tested, with the most suitable being finally adopted.
- The work may also be seen as a step forward in the fields, where the achievements of NLP are applied (e.g., robotics, automatic navigation and geoinformation systems).

1.3 Document Outline

The rest of the document is organized as follows: Section 2 describes the fundamental concepts used in the area, as well as the previous achievements in the fields of SRL and SpRL. Section 3 describes the modifications and the improvements introduced to tackle the SpRL task. Section 4 is dedicated to the evaluation methodology and lists the obtained results. Section 5 draws a conclusion and also mentions some problems we find interesting for the eventual future investigations.

2 Concepts and Related Work

This section formally introduces the Spatial Role Labeling task, describes fundamental machine learning concepts and presents related work on Semantic Role Labelong and Spatial Role Labeling.

2.1 Spatial Role Labeling

The SpRL task assumes automatic labeling of words or phrases in a sentence with a set of spatial role tags. The basic roles are the spatial indicator, the trajector and the landmark. Secondary spatial roles reflecting motion, path, direction, distance can be further introduced.

The *trajector* is the entity (i.e. object, person or event) whose location or motion is of relevance. The *landmark* is the reference entity in relation to which the location or motion of the trajector is determined. The *spatial indicator* is the spatial pivot that is assumed by each relation. The *motion indicator* represents a spatial role conveying the motion of

the trajector of a spatial scene along a path [1]. The *path* is a spatial role label that is assigned to a word/phrase that denotes a path along which the trajector moves. The direction denotes an absolute or relative direction of motion, or a spatial arrangement between a trajector and a landmark. 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.

In the SemEval-2013 [1] 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.

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.

2.2.1 POS Tagging as a Simple Sequence Tagging Task

The objective of a sequence tagging task is to build a model that would map a sequence of words in a natural language onto the corresponding tag sequence. One of the simplest *sequence tagging tasks* for NLP, where machine learning was introduced, is part-of-speech (POS) tagging. The output tags receive the interpretation of the parts of speech that the input words belong to. The model is trained on a corpus of examples. A traditional way to address sequence tagging tasks is based on Hidden Markov Models (HMMs).

2.2.2 Hidden Markov Models and the Generative Approach

N-order HMMs are stochastic models used to describe randomly changing systems. They possess the Markov property, i.e., the future state depends only on the current state and on N previous states. The output sequence, and not the sequence of states, through which the system passes, is directly visible. Each HMM is characterized by 3 probability distributions: the distribution $P(y_i)$ over the hidden states, the transitional distribution from one hidden state to another P($y_i|y_{i-1}, ..., y_{i-n}$) and, finally, the distribution $P(x_i|y_i)$ of the observed symbols over the hidden states.

To develop HMMs, generative approach is usually used. It focuses on determining the joint probability of observing the input sequence $x = (x1...xn)$ paired with the output tag sequence $y = (y1...yn)$. This joint probability can be further decomposed in a way that takes its source in the conditional probability definition.

2.2.3 Linear Classication Models and the Discriminative Approach

In the discriminative approach, the conditional probability P(y—x) is modeled directly, with the model directly implementing the function:

$$\arg\max_{y \in Y} \mathrm{P}(y|x) \tag{1}$$

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

$$\arg\max_{y \in Y} \Sigma_{i=1}^{N} \theta_i f_i(x, y) \tag{2}$$

Here, $f_i(x, y)$ is the i-th *feature*, and θ_i is the corresponding *feature weight*. The classier further attributes the sample to the class that has the biggest value of the scoring function.

2.2.3.1 Perceptron

: The *Perceptron* is the simplest representative of the family of linear classifiers. A standard Perceptron is a binary classifier, which means that it is only capable of separating the samples into two groups. To perform classification, the Perceptron model first computes the $\mathrm{Z}(x)$ function, such that:

$$\mathrm{Z}(\boldsymbol{X}) = \Sigma_{i=0}^{m} w_i x_i + b \tag{3}$$

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*, with non-linear functions, such as the *sigmoid* being most frequently used.

Each round of perceptron training is represented by the following iterative procedure [2]:

- 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.

2.2.4 Conditional Random Fields

Linear-chain Conditional Random Fields models (hereinafter CRF models) combine the ability of modeling compound outputs and that of taking into account the multiple features of the neighbouring elements of the input sequence. The basic idea of CRF models is to attribute a label (i.e., a score) to the possible tag sequences. This is done leveraging the inventory of feature functions. A feature function $f(x, i, l_i, l_{i-1})$ is a binary function, which represents a statement regarding a pattern in the input data [3]. The *weights* are determined 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.

The scoring is performed using the following formula:

$$\mathrm{s}(l|x) = \Sigma_{j=1}^{m} \Sigma_{i=1}^{n} \theta_j \mathrm{f_j}(x, i, l_i, l_{i-1}) \tag{4}$$

Being a 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. Secondly, CRF do not impose any restrictions on the feature functions weights, while in the case of HMM the normalization constraint should be met, because the parameters $P(x|y)$ are interpreted as probabilities [3].

2.3 Related Work

This section presents previous studies addressing the SRL and SpRL tasks.

2.3.1 SENNA and NLPNET

SENNA is a system designed for tackling a number of NLP tasks, namely SRL, POS tagging, named entity recognition and chunking. The creation of SENNA pursued two main goals. Firstly, it was desirable to avoid task-specic engineering, based on employing linguistically motivated features. Secondly, the system was conceived to offer state-of-the-art performance in the addressed tasks [4].

SENNA is based on a *multilayer neural network* architecture. Task-specific word representations are learnt step-by-step by successive network layers, with the words of a sentence serving as input for the first network layer. The backpropagation algorithm is used to train the discovered features. A word is represented by a set of features, while each feature is characterized by a feature vector from the *lookup table*.

There are two types of network conguration used by SENNA. The window approach is sutable 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. The convolutional approach provides for modeling semantic inter-dependencies between the words in the whole sentence. It is deemed useful to deal with complex NLP tasks, such as SRL. 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.

There are several re-implementations of the SENNA system, which can be adopted to deal with the SpRL task. The NLPNET system [5] is one of them. The peculiarity of NLPNET, in comparison with SENNA, is the use of a 3-step approach, with three separate models being employed for each of the steps. The steps are spatial pivot identication, argument identication and argument classication.

2.3.2 The Tensor-based Approach to SRL

The tensor-based approach to SRL is focused on determining the semantic dependencies for each predicate, that belongs to the set of the analyzed semantic pivots in a sentence. The input for the system is a syntactic dependency tree y_{syn}, produced by a state-of-the-art parser. The system leverages the inventory of scoring functions, which should reflect high-order interactions between the predicate, the argument paths and the associated role labels. The traditional feature scoring function is combined with a tensor-based scoring function.

The traditional feature scoring function is the inner product of the parameters vector $\boldsymbol{W}$ and the feature vector (x, y_{syn}, z_{sem}). The tensor-based scoring function characterizes each semantic arc (p, a, r), where a is the argument and r is the corresponding semantic role label. The cross product operation is applied to the atomic feature vectors, associated with the four types of information, concerning the predicate token, the argument token and some features characterizing the dependency label path, that connects the predicate with its arguments.

The objective of the *learning process* is to optimize the set of the model parameters, which is done via an iterative parameter tuning procedure.

2.3.3 KUL

The KUL system represents a *pipelining approach* to SpRL [6]. First, spatial pivots are learnt in the input sentence, leveraging a discriminative or a generative classier. The result is fed to the *spatial role labeler* afterwards, and the spatial arguments (i.e., the trajector and the landmark) are identied. The semantic parse tree serves as auxiliary input data for the task. Both spatial pivot identication and argument identication leverage a number of linguistically motivated features.

2.3.4 UTDSpRL

The UTDSpRL system for SpRL was conceived as a competitor to the pipelined approach, represented by the KUL system [7]. According to the authors of UTDSpRL, each of the two main spatial arguments cannot be identied without any additional knowledge about the other one. The authors' idea consists in a joint approach, that provides for deciding, whether a candidate triplet *(trajector, spatial indicator, landmark)* actually reflects a valid spatial relation in terms of real-life spatial semantics. A high-recall heuristics is employed to tackle the issue of recognizing the objects, capable of taking part in spatial relations. All possible combinations of arguments are then fed to a binary classier, which makes a joint decision.

The system uses a large number of manually engineered features and selects the most important ones via a special procedure.

3 Modications and Improvements for Spatial Role Labeling

Spatial Role Labeling requires several steps. Each of these steps implies machine learning and results in a distinct model, based on a multilayer neural network. These steps are *spatial pivot identication*, *argument identification* and *argument classification*.

Due to the fact that NLPNET was not originally designed for SpRL, nor was the SpRL formalism meant to be used by a system like NLPNET, it was found useful to pre-process the training data before feeding it to the model training algorithms.

3.1 Network Architecture

The advantages of using two distinct neural networks for argument identification and argument classification is that it provides for using the most appropriate network configurations for each of the steps. It also allows for using features that are mostly appropriate for each of the sub-tasks and avoiding error propagation.

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. Due to the relative simplicity of the sub-task, SENNA-style window approach with a

multi-layer perceptron (MLP) neural network is employed. The assumption is that the word's tag depends on its N neighbours. A matrix of word embeddings corresponding to the window, is formed and later on processed by a feed-forward neural network. The vectors of the matrix are concatenated and fed to the subsequent network layers. The last layer outputs the score of the word in the middle of the window being a pivot or not.

The model for *argument identication* determines the spans of the subordinate arguments for every spatial pivot (i.e., the network outputs scores for the respective IOBES tags), while the model for *argument classication* would attribute the appropriate tag to each of the discovered spans. The neural networks are quite similar, although certain minor differences exist. Both are *convolutional networks* relying on feature vectors updated in the course of the training process. The networks perform *convolution*, i.e., the convolutional neuron layer applies a weighted sum to the feature vectors of the overlapping windows. 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 *argument classification* sub-task is considered simpler, so that no hidden layer is used after the convolutional layer.

The representation of each word in the sentence should be complemented with two additional features. One of them reflects the relative distance from to the word that is currently being tagged. The other one is responsible for the relative distance to the currently considered spatial pivot, as the process is repeated for all of the previously determined predicates.

The *transitional score matrix*, used for *argument identification*, contains the scores for each tag pair, that reflects the possibility of encountering one tag immediately after the other. The elements of the matrix are updated at the training process, just like the the word representations and the neuron weights.

In the case of *argument identification*, the final score, attributed to a tag sequence at the inference phase, is the sum of the scores for each of the labels, assigned to every word, and the scores of observing one tag after the other, which is fetched from the transitional score matrix. To prevent from overfitting, *argument classification* uses no transitional score matrix. In both cases, the most adequate tagging sequence is determined with the help of the *Viterbi dynamic algorithm* procedure.

3.2 Network Training Parameters

The usage of a separate model for each of the three sub-tasks, comprising SpRL, provides for a finer tuning of the parameters of the training process. 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. The learning rates for the three sub-tasks equaled to 0.001, 0.001 and 0.01 respectively. We set the number of rounds equal to 1 for spatial pivot identification, 15 for argument identification and 13 for argument classification, with the last 10 rounds of the last sub-task having been performed at a 10 times lower learning rate to prevent overfitting.

Multidimensional word2vec embeddings, was the ultimately adopted type of embeddings due to their accessibility, robustness and the big size of the text corpus used to produce them.

3.3 Input Data Format for System Training

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 conventions be met. Data files should contain sentences, separated by a blank line. Each token that is part of a sentence is characterized by an ordered predefined set of features.

However, both of the gold annotation sub-corpora, used for system training, were made available in the SemEval-2013 shared task format [1], i.e., they stuck to a specific XML format.

It was thus necessary to create a script for converting from the SemEval XML format to the CoNLL annotation scheme.

3.4 Training Data Pre-processing

Experiments were made with two different datasets, respectively containing (i) only stating spatial relations, and (ii) a mixture of both static and dynamic spatial relations. Different challenges were addressed in the pre-processing of these datasets.

3.4.1 Static Relations

The phrases, containing multiple spatial tuples having the same spatial pivot, were the main bottleneck for the case of static relations. The system was capable of discovering only one of the bunch of such tuples, leading to a moderate value of recall.

The idea was to create one *compound relation* out of various relations, sharing one spatial pivot. A compound relation is a tuple, containing a spatial indicator and various *co-trajectors* and/or *co-landmarks* subordinate to this spatial indicator.

With the addition of more arguments to the spatial relations, new spatial role classes had to be introduced. *Trajector-1*, *trajector-2*, *trajector-3*, *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*.

It was necessary to represent the newly introduced 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. After the system was trained to learn the relations from the *modified annotation file*, it would be able to produce the desired output at the inference phase.

The output, obtained at inference, that followed the *modified formalism*, should be later converted back to the original format via an *inverse transformation*. This transformation consisted in spawning several simple spatial tuples out of one compound tuple, determined by the system.

3.4.2 Dynamic Relations

The addressed 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 and detected at the pivot identification stage. The format required that each tuple could only have one role, marked as the pivot. Consequently, we had to ignore the pivotal nature of either a spatial or a motion indicator for dynamic relations with spatial indicators. The relations sharing the same pivot could also be one of the issues, just like in the case of static relations.

The first of the alterations affected those tuples, 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.

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, 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*.

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, previously discussed for the case of static relations.

3.5 Programming Technology

The scripts, aimed at improving the quality of the models, 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.

This choice gave us an opportunity to use a number of Python packages for advanced mathematical operations. These packages included NumPy, NLTK and Gensim, which is a useful tool for tackling various NLP tasks in terms of dealing with big-sized training data.

4 Experimental Evaluation

This section describes the experiments carried out in the course of this work, the datasets used for system training and evaluation and the leveraged evaluation metrics and scheme. It also presents the experimental results.

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. The first part is composed of descriptions of photos, taken by tourists and contains only static spatial relations. The second part is composed of descriptions of locations situated at the intersections of certain latitudes and longitudes and contains both static and dynamic relations. The corpora had been previously annotated according to the SemEval annotation scheme, with the MAE tool [8] being employed. The annotations are placed in les, which can be processed as well-formed XML documents.

4.2 Models under Evaluation

For each of the two studied cases (i.e., static relations and dynamic relations), we trained 4 distinct models. For every group of four, two models were trained and evaluated on the same corpus, while the two others were tested on previously unseen data. The first model for each pair did not leverage any input data format transformations for the training phase. Conversely, the other model of every pair used the proposed transformations to boost the values of the performance indicators.

4.3 Evaluation Methodology and Evaluation Metrics

We used the two-step evaluation methodology, proposed at the SemEval-2013 shared task. On the first step, the system's capacity for individual role labeling was evaluated, while on the second stage complete spatial role labeling capabilities of the models were put to the test.

By correctly determined individual roles we mean the correctly identified roles, which could be possibly part of incorrectly determined tuples. The evaluation scheme is *span-based*. 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*. For complete tuple identification, each instance 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.

The *precision*, *recall* and the *F1* indicators were employed to measure how good the system's performance was regarding the criteria used for each of the steps of the evaluation process.

4.4 Evaluation Results

We further present the results for the models that were evaluated on unknown data (i.e., the datasets used for training and evaluation were different datasets). The first model of each pair did not apply any pre-processing to the training data, while the second leveraged the proposed input data format transformations.

TABLE 1
Static Relations. Performance Evaluation Results for the First Model

Spatial Role Class	Precision	Recall	F1
INDIVIDUAL ROLES	0.891	0.579	0.702
COMPLETE TUPLES	0.379	0.282	0.323

TABLE 2
Static Relations. Performance Evaluation Results for the Second Model

Spatial Role Class	Precision	Recall	F1
INDIVIDUAL ROLES	0.857	0.662	0.747
COMPLETE TUPLES	0.500	0.425	0.460

TABLE 3
Dynamic Relations. Performance Evaluation Results for the First Model

Spatial Role Class	Precision	Recall	F1
INDIVIDUAL ROLES	0.749	0.174	0.282
COMPLETE TUPLES	0.317	0.087	0.137

TABLE 4
Dynamic Relations. Performance Evaluation Results for the Second Model

Spatial Role Class	Precision	Recall	F1
INDIVIDUAL ROLES	0.719	0.242	0.362
COMPLETE TUPLES	0.250	0.108	0.151

We can see that both for static and dynamic relations, the proposed transformations provide for obtaining models that are characterized by a higher value of recall and a slightly lowered precision. Nevertheless, the F1 score improved both for static and dynamic relations after the pre-processing was applied to the training data.

5 Conclusion

The approach to SpRL, proposed in this MSc thesis, uses a multilayer convolutional neural network architecture and contrasts with the methods that had been leveraged to create the existing systems for the task [6], [7], [9]. Instead of relying on linguistically motivated feature engineering and heuristics, it uses word embeddings as the main features. One notable advantage of word embeddings is the fact that they are trained in an unsupervised fashion. Furthermore, the convolutional network is capable of discovering new relevant features during the training process. We can state that making use of the combination of word embeddings and a convolutional neural network is in line with the tendencies that aim at reducing the size of the corpus used for system training.

The proposed input data format transformations aimed at adjusting the existing annotation scheme to be more effectively interpreted by NLPNET at the model training stage. For static relations, the proposed transformation provided for an increase in the F1 metric for complete tuple identication, which was of 42% for unknown data and of 30% for previously seen data. In the case of dynamic relations, the increase was of 28% for previously unseen data and of 6.7% in the case of previously seen data.

The main limitation of the solution is probably the fact that model training and inference require considerable computational power due to the big size of the corpus of pre-trained word embeddings.

Despite the interesting results, there are also many ideas for future work. These include:

- Classify the obtained spatial relations into regional, directional and distal.
- The solution for SpRL might still use some linguistically motivated features, such as POS tags, to ensure better results.
- Adopt the method, proposed by P.Kordjamshidi et al [10], that benefits from using visually informed embeddings of words (VIEW), induced on a large corpus of annotated images. The VIEWs may be concatenated with the used word2vec representations for words and be further loaded by the NLPNET system at the training stage. This might be a promising way to reflect real-life spatial semantics and constitute an alternative to using heuristics.
- In a more remote perspective, we can also experiment with the high-order low-rank tensor approach, that has managed to achieve impressive results for SRL.

References

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

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

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

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