Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Ilana Rampula

Semantic Relation Extraction from Unstructured Data in the Business Domain

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: RNDr. Pavel Pecina, Ph.D.
Gosse Bouma, Ph.D.

Study programme: Master of Computer Science

Specialization: Mathematical Linguistics

Prague 2016

I would like to thank my supervisors, Pavel Pecina in Prague and Gosse Bouma in Groningen, for their guidance and insight throughout the thesis. I am grateful for the unique opportunity of working on real business data. I would like to thank everyone who made it happen. Very special thanks go to my dear friend Feraena Bibyna, whose help and support had a great impact on the completion of this thesis. Finally, I would like to thank my family for their love and endless support.

I declare that I carried out this master thesis independently, and only with the cited sources, literature, and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague date 13.05.2016 signature

Název práce: Extrakce sémantických vztahů z nestrukturovaných dat v komerční sféře

Autor: Ilana Rampula

Ústav: Ústav formální a aplikované lingvistiky (ÚFAL)

Vedoucí diplomové práce: RNDr. Pavel Pecina, Ph.D. & Gosse Bouma, Ph.D.

Abstrakt:

V posledních letech se využití textové analytiky v komerční sféře postupně stává významným tématem pro vědecké a praktické aplikace. Zaměřili jsme se na určování vztahů mezi entitami z dat dodaných partnerskou společností. Analýza textu z této sféry ale vyžaduje jiný přístup: počítání s nepřesnostma a specifickými atributy. V této práci jsme se rozhodli ukázat využití dvou metod pro určování vztahů: tzv. Snowball systém a Metodu vzdáleného dohledu (z angl. Distant Supervision), které jsme přizpůsobili pro dodaná data. Dané metody byli implementovány pro využití strukturovaných a nestrukturovaných dat z firemní databáze.

Klíčová slova: Získavání informací, Určování vztahů mezi entitami, Textová analytika, Distant Supervision, Snowball

Title: Semantic Relation Extraction from Unstructured Data in the Business Domain

Author: Ilana Rampula

Institute: Institute of Formal and Applied Linguistics

Supervisors of the master thesis: RNDr. Pavel Pecina, Ph.D. & Gosse Bouma, Ph.D.

Abstract:

Text analytics in the business domain is a growing field in research and practical applications. We chose to concentrate on Relation Extraction from unstructured data which was provided by a corporate partner. Analyzing text from this domain requires a different approach, counting with irregularities and domain specific attributes. In this thesis, we present two methods for relation extraction. The Snowball system and the Distant Supervision method were both adapted for the unique data. The methods were implemented to use both structured and unstructured data from the database of the company.

Keywords: Information Retrieval, Relation Extraction, Text Analytics, Distant Supervision, Snowball

Contents

Introduction

Text Analytics in the business domain is a growing field both in research and in practical applications. It is not surprising that most of the commercially available tools for advanced analytics implement text analytics or data mining components[1]. Those companies usually include text analytics under the umbrella of Business Intelligence (BI). Business Intelligence is most commonly referred to as a collection of software tools and algorithms allowing a business entity to analyze their data and deduct information that would benefit their business needs. The tools for BI are expected to support all levels of business management and business processes. This kind of support can be only provided via in-depth data analysis, using all available data sources (Baars & Kemper, 2008). That said, up until recent years most of BI infrastructures were built on structured data alone, ignoring the huge quantities of unstructured information (Fan, et al., 2006). Structured data opposed to unstructured data is highly organized information, usually stored in a database with a predefined metadata and format. Structured data is composed of numerical data such as keys and scores. It can also contain string values with a known format such as names, emails, flags, and categories. This type of data is easily searchable. It can be processed directly and is usually self-explanatory. Unstructured data, on the other hand, has no apparent structure. It is very heterogeneous, as would be expected from natural language. It can be a challenging task to extract specific pieces of information from such data. Nevertheless, the constantly growing amount of data and the needs of the business entities to understand their data lead to an unavoidable shift in Business Intelligence approaches to the analysis of unstructured data, more commonly called Text Analytics. That is, text analytics is considered an inseparable part of understanding company's data. The outputs of text analysis can be used in business decision making or in operational processes. Marketing campaigns can be directed at specialized customer audience, offering the right products to the right clients. Business management relies on reports and dashboards when planning next steps, text analytics can provide more information on trends and preferences of the business customers. There are many more fields in

[1] For a review see http://www.predictiveanalyticstoday.com/top-software-for-text-analysis-text-mining-text-analytics/

the business domain that can use text analytics for their benefit, such as, email classification, summarization and sentiment analysis.

Out of the diverse ways to use text analytics methods in the business domain, we chose to concentrate on Semantic Relations Extraction (RE) between Named Entities (NE). RE is a way to represent unstructured data in a structured form. Such transformation could allow business analysts to use the volume of unstructured data in their models combined with already available structured data. Semantic Relations express a given quality between two (or more) entities. An entity, in this case, is a Named Entity such as a person, location, organization, date, etc. A relation can depict anything from the place of birth to the year of graduation or date of establishment. By extracting specific facts from the text, we actually convert subpart of unstructured data to structured data. Since the focus of big companies is clients' information, semantic relations between the client and another entity provide a clear added value. As mentioned before, the purpose of BI is to collect information for making better decisions. Any additional relevant piece of information about a company's clients could immediately improve the quality of the data the company holds.

Semantic Relation extraction usually consists of two main steps. Firstly, in order to extract Semantic Relations from unstructured data efficiently, the text has to be labeled with Named Entities. The process of automatically detecting and labeling Named Entities is called Named Entity Recognition (NER). Named Entity Recognition is not a trivial task. Fortunately, there exist ready-to-use tools for NER. For example in this work, we use the NameTag tool for annotating named entities in Czech (Straková, et al., 2013). The second step is to understand how a certain relation between named entities is expressed in the unstructured text. The methods for detecting relations in text vary on several levels. First of all, there are many ways to parse and analyze the text surrounding the extracted named entities. For example, one would have to decide whether to parse the text on a morphological or syntactic level. Choosing the features to represent the text has one of the greatest effects on the results of the extraction. In addition, the methods for RE vary on the level of supervision they require and the algorithm that implements the extraction. There are supervised, semi-supervised and unsupervised methods for RE. In the following chapters, the implementation from the phase of data preparation and processing through feature selection to the implementation of the methods themselves is described in greater detail.

The thesis presents two methods for RE – Snowball[2] (Agichtein & Gravano, 2000) and Distant Supervision (Mintz, et al., 2009). Supervised machine learning methods require significant investment into hand labeling training examples. Both methods counter the need for manual labor, each in their own way. The Snowball system is initiated using a number of seed examples that are input into an iterative process to extract more examples to be used for training. Distant Supervision takes advantage of existing databases that contain lists of named entities and the relations that hold between them. The database is then used directly as training examples, eliminating the need of manually creating training data. For the purpose of extracting RE from the business based corpora, both methods are modified to fit the domain.

The data used through the thesis was provided by an industrial partner. We have acquired the company's agreement to use a small part of their database as the dataset in this work. The relational database consists of client records, with several tables containing different detailed information related to the client. As in majorities of large companies, a significant part of the database is in the form of raw unstructured text. We were granted access to one of the tables in the database containing such data. Opposed to corpora containing only textual data, such databases are unique because all the information, structured and unstructured, is interlinked. Every piece of unstructured text is connected to a client profile and personal information. In this thesis, those links are used to enhance the process of relation extraction. On the other hand, we demonstrate how the extracted relations enrich the structured data. Given the data provided to us, we chose the *country-of-origin* relation serving as an example for the RE methods implementation.

Working with such data presented several challenges. The corpora are rather large and sparse, with many missing or wrong entered values. Even the structured data is rarely unified, having variations in the same categorical value. The unstructured data contains domain specific abbreviations and misspellings. Additionally, the data is in the Czech language, bringing complexity to the task, as Czech is a morphologically rich language.

As mentioned above, there are numerous tools on the market focusing on text analytics for the business domain. To the best of our knowledge, relation extraction from unstructured data as presented in this work was not implemented by any of those

[2] Not to be confused with the Snowball stemmer.

tools. Furthermore, the research of relation extraction is concentrated on different domains, such as Wikipedia, news feeds, and social media. Relation extraction from the business domain's unstructured data requires a different approach, counting with irregularities and domain specific attributes. We believe that the approach provided in this work is novel, particularly due to the unique data and the modification of the methods implemented here. Moreover, the methods depicted here are suitable for the extraction of a variety of relations, depending on the available database structure, type of information stored in it and the information the company is interested in extracting. For example, using a database containing bio-medical information, one can extract biomedical relations, between symptoms, medications, and effects using experiment descriptions as the source of unstructured data.

The structure of the thesis is as follows:

In the first chapter, a throughout background is given in several sections. The chapter includes an overview of text analytics in the business domain, RE definition and detailed description of RE methods, algorithms and evaluation metrics. The data description, preprocessing methods, and data analysis are given in Chapter 2. In this Chapter, we also present the practical challenges involved in working with real data and the linguistic challenges of analyzing Czech text. The modification and the implementation of the used methods are described in Chapter 3. The description of the experiments, the different feature sets used in them and the results of the experiments are depicted in Chapter 4. Chapter 5 is dedicated to concluding remarks and future work.

1 Background

1.1 Text Analytics in the Business Domain

In the growing world of unlimited data, enterprise companies are facing a serious challenge. The effort invested in documenting all available information regarding clients and other involved parties, market trends and competitors stands across losing the ability to understand the vast data at hand. By now, every big company understands that the key to gaining an advantage over its competitor's lies is data governance and analysis. Business intelligence (BI) is a range of solutions that enables companies to get a handle on its data through reports and analytical tools, where the main goal of a BI system is to prepare the grounds for making better, more intelligent decisions (Gangadharan & Swami, 2004). There is a variety of components that can be used by a BI system. In general, those components can be divided into two main types. The first is reporting and visualization, enabling an organized view of the data with specific stress on areas of interest. The other is analysis and data discovery, those components are delivering additional information not available in the existing data as is.

BI covers various fields, two of them are data mining and text analytics. Data mining is a process of finding patterns and tendencies in a large amount of data. Data mining uses many different methods for data discovery such as machine learning and statistics. Data mining usually refers to the analysis of structured data, while text mining or text analytics is the analogous process that deals with textual data. In a way, text analytics in the business domain is data mining on unstructured data. Technically, data mining and text mining are applying the same analytical functions on different data domains. Contrarily to structured data analysis, text analytics requires additional steps before the data is accessible to be used by a computer. Natural Language Processing (NLP) can help a computer to understand relevant pieces of information to be used in text analytics. NLP is used to convert raw text into structured features, in a way that those features can be used side by side with originally structured attributes. The purpose of both text analytics and data mining is to apply automatic methods in order to extract new information such as patterns, clusters, unique values or outliers, and other dependencies.

Data mining is capable of handling only a very limited section of the data resources of a company. In 1999, Dörre at al. estimated that about 90% of a company's data is of unstructured form. Probably, today, the distribution of structured and unstructured

data in a company's databases is tilted towards unstructured data even more. Data mining cannot deal with documents such as emails, letters, contracts and phone recordings (Dörre, et al., 1999). On the other hand, with platforms like Hadoop[3], it is possible now to store and process gigantic amounts of data of all types and sizes. Many BI software companies already realized the potential in the growing market of text analytics and there is a range of solutions available for Big Data users. Nevertheless, there is a vast range of opportunities to explore in the growing field. In our point of view, there is a place to integrate academic research achievements into practical applications. This thesis is an example of such integration.

Business entities are interested in a range of possible extracted information, from simple keyword detection to complex sentiment analysis. Moreover, text analytics can be used on a collection level to identify trends and classifications. More specifically, users of data and text mining are interested in various tasks like automatic classification of documents by topic, extraction of entities, events and concepts and identification of personal attributes and sentiments. A market survey conducted in 2014 by Alta Plana Corporation[4], which delivers business analytics strategy consulting, highlights the different use cases and approaches of text analytics in the business domain. According to the survey, text analytics is most useful in four industry groups: consumer-facing businesses, public administration and government, life sciences and clinical medicine, and scientific and technical research. All of the listed groups are interested in improving functional applications, such as search and customer service, mainly aimed at meeting operational needs (Grimes, 2014). In this thesis, the business entity providing the data is a customer-facing company. The main interest expressed by the company is in detecting personal attributes of clients, to be used in personalized campaigns and for other marketing purposes. Another use case of text analytics in this type of company is for data quality. Data Quality refers to the level of coherence and accuracy of the data representing entities in the real world. The extracted information can be used to fill in missing values in the database and to validate the existing values.

Theoretically, the tasks described above as part of text analytics could be performed by a person. There are several advantages in automated text analytics. Firstly, given a

[3] http://hadoop.apache.org/

[4] http://altaplana.com/index.html

large amount of text data, automatic processes are quicker than manual analysis. With modern technology, like Hadoop, it is possible to process very large document collections with parallel computing, many times faster than before. Secondly, humans are prone to errors, while computers are performing deterministically according to a given algorithm. Additionally, there are methods to counter over-fitting and generalization errors. Thirdly, human annotations are subjective, while computers have objective judgments. A real life example of the advantages brought by text analytics in the business domain is demonstrated in the following citation:

> "In 2001, Dow Chemical Co. merged with Union Carbide Corp., requiring the integration of 35,000 Union Carbide research and development reports into Dow's document management system. Dow partnered with ClearForest Corp., a commercial developer of text-driven business solutions, to help integrate the new combined document collection. Using technology it had developed, ClearForest indexed the documents, identifying chemical substances, products, companies, and people for inclusion in the combined database. Dow was able to add more than 80 years' worth of Union Carbide research to its information management system and approximately 100,000 new chemical substances to its registry. When the project was complete, Dow estimated it had spent some 3$ million less than it would have if it had used its own methods for indexing documents. Bow also estimated it had reduced the time it would have spent sorting documents by 50% and data errors by 10%-15%." - (Fan, et al., 2006).

The case study of ClearForest and Dow Chemical Co. is only a tip of the iceberg of what text analytics can accomplish in a variety of scenarios.

In the following sections, we demonstrate how with one text analytics task it is possible to achieve improved data quality and gain insight into a selected personal attribute of a client. The attribute is extracted in the form of Semantic Relation between the customer and the entity representing the attribute. Subsequently, the extracted values are brought back into the database to be used for integration in the company's predictive analysis or directly in various operational functionalities.

1.2 The Task of Relation Extraction

In the previous section, we described the goals and tasks of text analytics in the business domain. As mentioned above, in text analytics, various methods have been implemented for structuring unstructured data. Semantic Relations Extraction is an example of such a method. Given a raw text, the algorithm is expected to detect bits of relevant information, in a way that enables the user to populate a relational database with the outputs, add the extracted values to analysis algorithms or merely correcting the existing data values.

Semantic relations represent facts about named entities. For example, in the sentence *Prague is the capital of Czech Republic*, there are two named entities <Prague, Czech Republic> with a relation {X is the capital of Y}. Other entities can belong to the same relation, e.g. <Paris, France>. Moreover, this relation can be expressed in a variety of ways:

- *Prague, Czech Republic's capital is also the historical capital of Bohemia.*
- *Prague is the 15th largest city in the European Union, and it is the capital of the Czech Republic.*

A relation is defined as a set of entities $\{n_1, \dots n_i\}$ where the elements of the set are in a given relation R. For binary relations, the set is an ordered pair $< n_1, n_2 >$ with a predefined relation between the two entities in the pair. In our context, $< n_1, n_2 >$ are two named entities of the same or different type (i.e. person, location, date, etc.). A relation can be defined using a pattern. For example, the pattern $\{n_1\ is\ the\ father\ of\ n_2\}$ is a relation between two named entities, both of the type Person, expressed through the context words between the two named entities. We would say that n_1 and n_2 are in relation R if the relation holds, i.e. n_1 really is the father of n_2 (Bach & Badaskar, 2007). This relation can be expressed in many other ways, corresponding to additional patterns. The patterns can be used to extract entities in the given relation. By simply comparing the sentences in the dataset to the pattern, we can find new pairs belonging to this relation.

The first step before approaching a relation extraction task is to detect the entities involved in the relations. In named entity recognition, a tool is trained to distinguish tokens referring to named entities from other tokens. Every language marks differently proper names, for example, by capitalization, special cases or syntactic constructions. Therefore, a named entity recognizer has to be implemented to use those language

specific marks. Generally, named entities are names of people, places, organizations times, and dates. The NER system not only identifies that a certain token is a named entity but also labels it according to the type it belongs to. This enables generalization over different types of named entities. For example, one can deduct patterns surrounding proper names of people and differentiate them from the patterns surrounding locations.

NER holds two challenges on top of the actual recognition. Firstly, same named entity can be expressed in different ways. For example *United States of America* refers to the same named entity as *USA*. When extracting relation between named entities, it is important to relate the numerous different occurrences of NE to the same real entity in the world. Secondly, not all mentions of named entities are transparent. Take for example the sentence: *John said to Marry that he was happy to see her.* An NER system will recognize correctly two named entities of the type Person in the sentence – *John* and *Marry*. But there are two more mentions of people in the sentence, of the form *he* and *her*. Relating the pronouns to the correct names mentioned previously in the text is a sub-task of co-reference resolution called anaphora resolution. Ideally, we would want to consider the pronouns that refer to NE when extracting semantic relations. In this work, we do not perform anaphora resolution.

After the detection of named entities is completed, we can extract the relations between them. As was noted by Craven and Kumlien (1999), extracting relations between named entities is actually an information extraction task. The goal of an information extraction task is to correctly identify text instances that express the required information without falsely marking text instances that do not. In the case of relation extraction, the required information is a relation between two named entities expressed through the context surrounding them. Each text instance in the corpora (a sentence, record or paragraph) can be classified as either relevant or irrelevant with respect to the required information. Namely, either containing information regarding the desired fact or not. Instances containing the information are considered positive examples and instances missing it are marked as negative examples (Craven & Kumlien, 1999). Here, we use a similar approach to RE, defining it as a binary classification task.

There are several options how to classify the text instances. One of the options is to hand code rules or patterns that would deterministically classify the texts as relevant or irrelevant. In the previous section, we noted the advantages of automatic methods

as opposed to manual methods. Additionally, constructing well-performing extraction patterns is a hard task that can be performed only by an expert. It requires a deep understanding of how certain relations are expressed, and a significant amount of time to capture all possible cases and exceptions. In the next section, we present methods for automatically identifying relations.

1.3 Methods

1.3.1 Supervised Methods for Relation Extraction

One group of methods for automatic classification is based on machine learning. Machine learning algorithms differ in the type of supervision they require. In supervised machine learning, some instances from the collection of texts have to be manually labeled. The instances are sections of text with one label per instance. A sentence is usually taken as an instance, however, bigger or smaller chunks of text are possible as well. After the training data is labeled, a model is trained on the labeled examples. This model can then be used for classifying new unlabeled instances. Bringing this back to our context, only some of the text instances actually express a relation in a given corpus. Our goal is to find as many instances containing related pairs as possible, without extracting false examples. If we would come to implement supervised machine learning algorithm for RE, we would need to read through the text and decide which instances are positive examples of a relation and which instances are negative examples. Usually, we would need to have a substantial amount of labeled examples, where the concrete number of training examples depends on the task and data at hand. Such a framework is called supervised machine learning since we supervise the learning process with the labeled examples.

In order to classify the units of text using a machine learning algorithm, we need to convert them into a structured representation usually in the form of a feature vector. Namely, we need to extract the text features that may help us to determine if the text expresses the relation we are interested in. The features can be anything from word forms in the text up to syntactic relations. The features can be also constructed from other related information, such as the length of the instance, the origin of the text instance (i.e. web page) or any other metadata connected to the particular data instance. Vector representation helps us to count how many times each feature appeared in the text instance (or in the whole collection). We can then easily compare vectors

representing different data instances and present the difference between the vectors in a numerical form, for example using the cosine measure. Figure 1. describes the process of supervised machine learning for relation extraction.

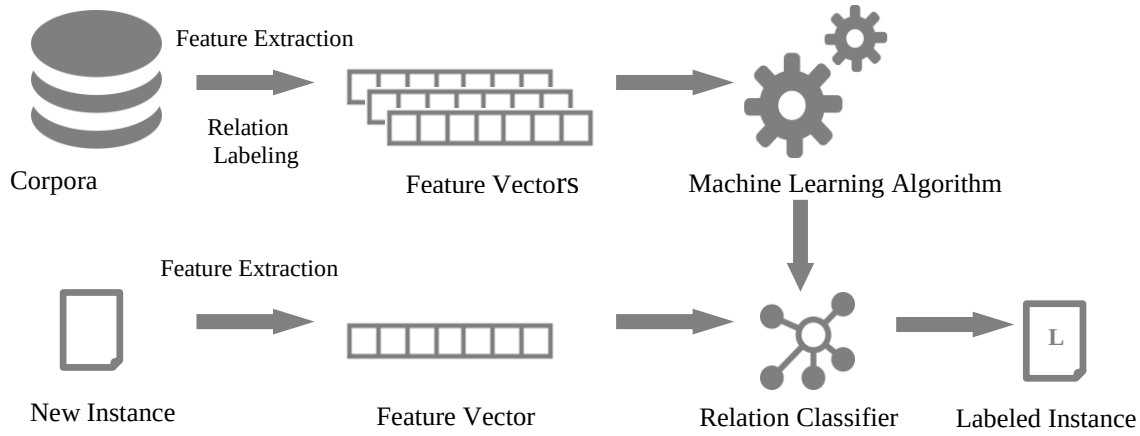


Figure 1. Supervised Machine Learning for RE

The main drawback of supervised methods is the amount of manual labor involved in labeling the training data. Moreover, in order to extend the method to the extraction of new relations or a new domain, one would need to create new training data and the model needs to be retrained on newly labeled data. The supervised methods might not scale well with the increase in the amount of data. If the added data is not similar to the training data, the methods performance might decrease (Bach & Badaskar, 2007).

In semi-supervised methods, the instances are not labeled with the relations for training, but some other information is given to the machine learning algorithm to partially guide the learning process. Semi-supervised methods usually rely on existing databases with a predefined ontology. Since this kind of reference database is not always available, some researchers suggested open relation extraction – an unsupervised approach (Banko, et al., 2007). Unsupervised machine learning algorithms are implemented to group unlabeled examples according to similarity, building on the fact that instances expressing the same relation would share similar feature values.

Next, we present two semi-supervised methods that are proposing two different ways of dealing with the lack of labeled data. In the following section, the Snowball system is described – a bootstrapping approach which uses a number of seed examples to iteratively extract new entities. The Distant Supervision framework follows. This

method introduces weakly supervised examples to be used as training material for a machine learning algorithm.

1.3.2 The Snowball System

Snowball (Agichtein & Gravano, 2000) is a relation extraction system that uses only a number of training examples, usually referred to as seeds. Contrary to the supervised machine learning approach, the only manual step in Snowball is the construction of the seed examples. From those training examples, the system builds relation extraction patterns, using the context surrounding the named entities in the initial seeds. The patterns are then used for extracting new entities with the same relation from the text. The process is iterative. Namely, the extracted entities can serve as the seeds for extracting new patterns. Essentially, the Snowball system is using a bootstrapping technique. It requires only a hand-full of labeled examples to initiate the process of relation extraction. In bootstrapping, the system initiates with a number of examples and uses the systems output to create more training examples to be used in the next round.

The Snowball system was built on the ideas introduced by Brin and his system the DIPRE (Brin, 1999). DIPRE stands for Dual Iterative Pattern Expansion. Brin extracted relations from a collection of HTML documents. DIPRE was optimized to extract relation from the web. It counts with repetitive occurrences of entities in a similar context. Similarly to the Snowball system, DIPRE requires from the user to input a small number of training examples to start the iterative process. DIPRE system does not utilize named entity recognition. Instead, the user has to come up with a regular expression that would represent a pattern the entities must match, for example, capitalization or the length of the name.

The second stage of the extraction is finding the entities from the seeds in the texts. Depending on the context words around the entities, DIPRE builds patterns that are expected to match the actual expression of the relation in natural language. First, the DIPRE algorithm converts the sentence with the detected entities into a 7-tuple: $< e_1, e_2, order, URL, left, middle, right >$. The 7-tuple consists of the two entities involved (e_1, e_2), the order in which the entities occur in the sentence, the URL of the document in which the entities were found, and the context words on the left, right and between the entities in the text. Next the 7-tuple representations are converted to patterns. Not all 7-tuples become a pattern. Instead, the 7-tuples are collapsed into the

most representative pattern. The pattern is converted to a 5-tuple, containing the order of the named entities, the URL prefix, the context words between and on the left and right of the named entity $< order, url\ prefix, left, middle, right >$. The order and the middle are matched in the grouped 7-tuples, and the URL prefix, right and left values are the longest common ground in the grouped occurrences. Brin limited the patterns to be supported by more than one seed and required the 5-tuple to have non-empty values. The constructed patterns can be used for finding new entities with the same relation. From this point the algorithm recursively repeats the step, using the extracted entities as the new seeds for the process.

To summarize, DIPRE follows the steps below:

1. Find seed examples in the text.
2. Construct patterns based on the text.
3. Find new pairs in the text using the patterns.
4. Repeat from step 1.

Coming back to the Snowball system, Agichtein and Gravano (2000) introduced two main modifications of the DIPRE algorithm. Firstly, the pattern extraction algorithm was modified to use named entity recognition, thus the patterns are different in the Snowball system. Secondly, Snowball uses an evaluation method to filter out weak patterns to prevent falsely extracted entities.

The pattern generation process in the Snowball system starts with finding the seed entities in the text, where they appear closely to each other. The seed patterns are prepared from the context around the seed NE. This step is the same as in the DIPRE algorithm. Next, the text is analyzed and tagged with named entities. NER helps to avoid ambiguous patterns. Figure 2. illustrates a simplified Snowball process. In Figure 2, one of the patterns is highly ambiguous: *<…PER from ORG…>*. Without NER, the pattern would extract all strings of the form *<NE1 from NE2>* (i.e. extracting both *"Bob from Florida"* and *"Bob from Vodafone"*). The snowball algorithm only considers sentences that contain a pair of named entities, ignoring sentences like *"the book from the shelf"*. DIPRE, on the other hand, used a rule-based approach, for

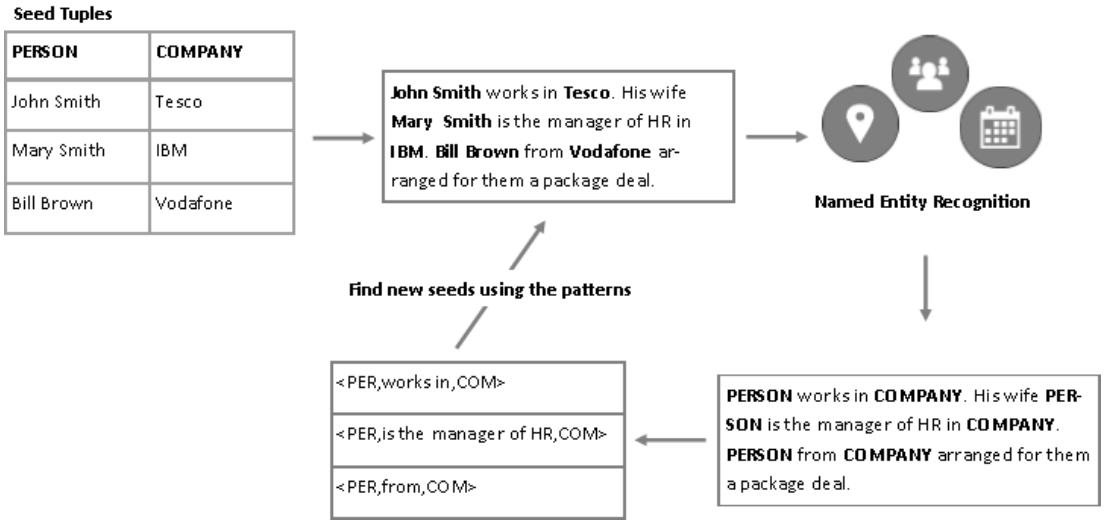


Figure 2. The Snowball System

example, limiting the entities to capitalized strings. Needless to say that a state-of-the-art NER has a higher accuracy in detecting named entities.

Additionally to NER, the Snowball system uses a more flexible representation of the context surrounding the named entity pairs. Instead of using the words on the left, right and middle in the pattern to find a precise match, the system represents the context in a vector-space fashion. The left, right and middle (LP, RP, MP) are represented as weighted vectors associated with terms. Thus, the patterns are flexible enough to capture minor variation in word order and additional words or punctuation. The patterns are constructed with the intention of capturing the majority of entities in the relation without catching invalid pairs of entities in the process. In other words, the system aims at both high precision and high recall[5].

Matching of patterns to text segments requires the texts to be converted to the pattern format as well. For every text instance that contains both named entities in question, the system generates a 5-tuple representation to be matched to the pattern. As mentioned above, the context in which the NE appear is converted to three vectors. The left and right vectors (LT, RT) are constructed by taking the weights of the terms on the left and on the right, within some predefined distance from the left and right NE. The middle vector (MT) contains the weights of the terms in between the NE. The weights of the terms are their frequencies in the respective context. The vectors are normalized and scaled by a factor of relative importance. According to Agichtein & Gravano (2000), the middle context contains the most informative features. Therefore,

[5] Precision and recall are described in the Evaluation Metrics section of this chapter.

the middle vector is scaled to have higher weights. The Snowball system compares the text instances to the pattern by summing up the corresponding context vector products.

$$Match(P,T) = LP \cdot LT + RP \cdot RT + MP \cdot MT$$

The Match score is calculated only if both NE are the same in the pattern and the text. Otherwise, the score is zero.

Generation of new patterns is accomplished by grouping text instances according to the named entities they contain. Using the Match function above, the algorithm calculates the similarity between the context vectors. For every pair of named entities, the instances are clustered with a threshold similarity. Every cluster represents a distinct pattern that expresses a relation between the two named entities. In order to finalize the representation of the pattern, each group of the three context vectors from the cluster is collapsed to its centroid. The result is a representative 5-tuple:

$$< left\ centroid, NE1, middle\ centroid, NE2, right\ centroid >$$

Those patterns are used in the next step of detecting new pairs of named entities with the same relation.

Extraction of new pairs of named entities from the text is performed in several steps. First, the system chooses text instances containing the entities that were labeled with relevant named entity tags. Then, the text segment is converted to a 5-tuple of the form $< LT, NE1, MT, NE2, RT >$. Snowball extracts the pair if the score calculated by the Match function is higher than a threshold (the value of the threshold is the same as in the previous step). The score is calculated for all relevant patterns and the pair is extracted if the 5-tuple is similar enough to at least one of the patterns. Next, Snowball filters out some of the extracted pairs based on an evaluation metric.

Evaluation is necessary for pattern generation. Often a pattern generated by one valid instance would extract invalid pairs, even if the pair is labeled with the correct NE. The reason for this would be the lack of valuable information in the context of those instances. To avoid generation of partially invalid patterns, the Snowball system chooses the patterns that are selective. To calculate the selectivity of a pattern, the system counts the precision of the pattern. Namely, how many times the pattern extracts correct pairs out of all extracted instances in which two named entities with the relevant tags occur. In this calculation, the manually constructed seed pairs are used to validate correct extractions. The system takes the first generated patterns from the seed instances and tries to extract new pairs with those patterns. The result is a list of newly extracted pairs for each pattern. Snowball then compares the list to the seeded

pairs. If it finds pairs in the list that only match with the first NE to the seed pair but not matches the other, the selectivity of the pattern is reduced. After the initial evaluation, only the best patterns are kept for the next iteration. NE pairs must be filtered as well. If a wrong pair is used in a new pattern generation, it will cause generation of incorrect patterns. The pairs are evaluated according to the selectivity and the number of the patterns that generated the pair. After the evaluation of the extracted pairs, the most reliable ones are added to the list of seed pairs. The extended list is used in the evaluation of the patterns in the next round.

In summary, Snowball is a relation extraction system that uses bootstrapping principles to minimize the need for manual labor. Additionally, the system is easily adaptive to new domains since the manually constructed seeds are the only requirement for training. For each new domain, the user needs to provide only new seed examples, without relabeling the text.

The method described above as well as the DIPRE method count with having multiple appearances of each pair of related named entities in the corpora. Thus, each pattern extracts only the entities that fit perfectly, without extracting false results. Collectively, all the patterns together should capture all possible instances of the relation. As shown later on, the assumption of redundant examples in the dataset does not hold for the task at hand. Meaning, in a limited dataset containing private personal information, there is only a handful of occurrences of each pair. Thus, the process had to be modified. We describe the variant implemented in this thesis in the Methods Chapter.

1.3.3 Distant Supervision

The Distant supervision approach was first introduced by Craven and Kumlien (1999). Their work focused on the extraction of biological information from various resources online that contain mainly unstructured data. The authors' goal was to map information from the unstructured knowledge sources into a database. Distant Supervision is a method that exploits existing databases for training data. In fact, related pairs found in the existing databases are directly used as "labels" for the text instances containing the pair. Craven and Kumlien (1999) coined such data instances weakly labeled. In their work, every sentence containing two named entities with a known relation between them is considered a positive training example. Such positive examples are then used in a text classifier. All other sentences are considered negative

examples. For classification, a Naïve Bayes classifier with simple unigram frequencies was used. In this implementation, each sentence is represented as a bag of words. The Naïve Bayes algorithm estimates the probability of a sentence expressing a given relation.

Mintz et al. proposed an extension of the method based on the ideas of Craven and Kumlien (Mintz, et al., 2009). In the implementation of their distant supervision method, they use a large semantic database – Freebase for supervising the relation extraction. Freebase is a database that contains relations such as *is-a* or *is-part* with corresponding examples of pairs in that relation. As in the original approach, Mintz et al. assumed that a sentence containing two words that exist in a semantic database expresses the relation of this pair. Several sentences can be combined to represent the way a given relation is expressed in the text. The representative sentences are converted to features to be used in a machine learning classifier of relations. The following diagram (Figure 3.) demonstrates the process of Distant Supervision as was implemented by Mintz et al.:

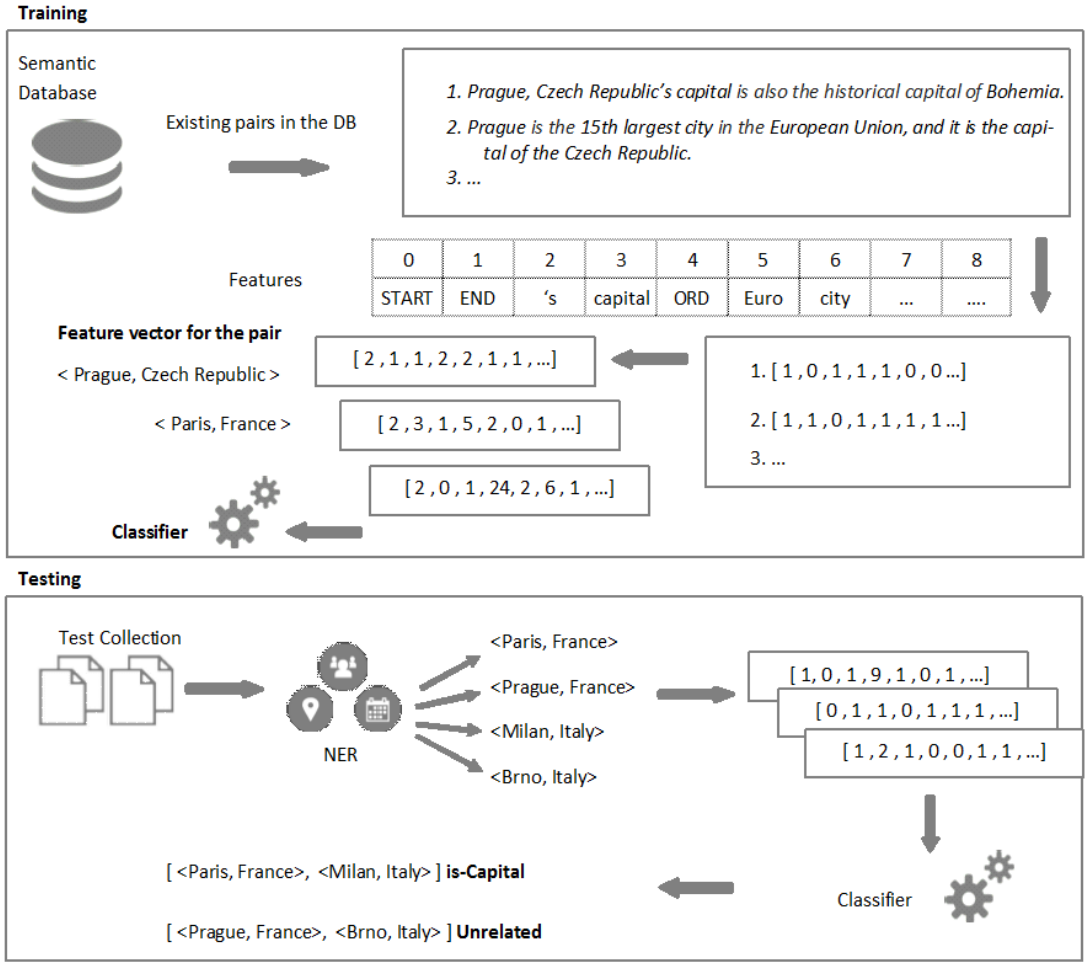


Figure 3. Distant Supervision

As seen in the diagram, the method is divided into two stages – training and testing. The first step of the training stage is named entity recognition. Out of the sentences labeled with named entity tags, the system selects sentences containing pairs mentioned in the Freebase database. Those sentences are processed for extraction of features. Next, if the same pair of NE appears in the text more than once, the features from the different sentences are combined into one feature vector. This step makes the feature vectors that are used as positive examples more informative. The feature vector constructed from different sentences will contain many different words that express the relation, making the probability that a new data instance with an NE pair in the same relation will be classified correctly. Note that this detail in the implementation is somewhat similar to the Snowball system – there we construct one representative seed vector from different seed examples to capture all the different features that express the relation. The training stage also includes a weighting stage that evaluates the features, giving lower weights to less productive features. In order to provide negative examples for the classification, more feature vectors are constructed from sentences with pairs that are not in any relation in Freebase. Based on the generated vectors, a multi-class logistic regression classifier is built. In the testing stage, the documents are labeled with NE. Then, features are extracted from all sentences containing two NE of the same type as in the relation of interest, for example <PER, ORG>. The system constructs a feature vector for every pair of named entities, from all the sentences containing the pair. The NE pairs are then classified with the logistic regression classifier. The output of the classifier is the name of the relation predicted for the named entities pair and the confidence score stating the probability of the predicted relation to be correct. The system was evaluated on held-out data from Freebase and on manually annotated data. The method uses several lexical features, such as the sequence of words between the two entities, POS tags of these words, the order in which the NE appear in the sentence, the words to the left of the first named entity and to the right of the second entity in a predefined window and their POS tags. The features that are entered into the feature vector are a combination of all of the features mentioned above. Each feature used by the classifier is then of the form: *<LWP NE1T MWP NE2T RWP>* where LWP stands for words and POS tags on the left, NE1T is the type of the first named entity, MWP is the middle context words and their POS tags, NE2T is the type of the second named entity and RWP is the right context words and POS tags. A conjunctive feature of this form limits the recall of the system and

improves its precision since all parts of the conductive feature from training should match the target data instance to be considered similar. As noted by the authors, this approach is only possible when dealing with large datasets. In small datasets, it could lead to too distinct features that appear in the data only once blocking the learning process.

The method implemented by Mintz et al. holds three main advantages. Firstly, the quantity of unlabeled data used in the method is unlimited. The number of pairs in the database does not need to be large in order to extract new pairs from millions of documents. Secondly, aggregating features from numerous source sentences improves precision dramatically. Thirdly, the method avoids domain specific constraints by using the unlabeled data directly (Mintz, et al., 2009). The same method can be applied using a variety of databases with different relations. However, such complex features as were constructed in their method can work only for a very large dataset.

The next section introduces the algorithms, tools and methods that we used to implement our variant of the Snowball system and the Distant Supervision method.

1.4 Algorithms and Methods

We present two frameworks for Relation Extraction. The Snowball system uses a bootstrapping algorithm for finding patterns for relation extraction. This method does not require a machine learning algorithm to learn the new patterns. Instead, it iteratively extracts additional examples using a vector space model. In the Distant Supervision paradigm, a classifier is used to mark new instances with relations. One of the ways to approach the classification task is a machine learning approach, implemented in the same way as a supervised method presented previously, only using weakly labeled training examples instead of manually labeled examples. In this section, we present the machine learning framework for Distant Supervision and the vector space model for the Snowball system.

1.4.1 The Machine Learning Tool – Vowpal Wabbit

For the implementation of Distant Supervision, we have chosen to use the Vowpal Wabbit (VW), an online machine learning tool (Langford, et al., 2007). The VW project is a learning system sponsored by Microsoft Research and before that by Yahoo! Research. It is an open source project for fast and scalable learning algorithms. VW is usually used for ad prediction, document classification, spam

detection, and more. The system includes several optimization algorithms with the baseline being sparse gradient descent (GD). VW has a combinatorial design, by adding command line parameters, one can change, for example, the features used, the loss function and the feature weights (see Table 1.).

Table 1. Vowpal Wabbit Design[6]

Vowpal Wabbit Design	
Format	{binary, **text**}
IO	{**File**, Pipe, TCP, Library}
Features	{**sparse**, dense}
Feature	{**index, hashed**} with namespaces
Feature manipulators	{**ngrams, skipgrams, ignored, quadratic, cubic**}
Optimizers	{online, CG, LBFGS} parallelized
Representations	{linear, MF, LDA}
Sparse Neural Networks	by reduction
Losses	{**squared, hinge, logistic, quantile**}
Multiclass	{One-Against-All, ECT}
Cost-sensitive	{One-Against-All, WAP}
Contextual Bandit	{Ips, Direct, Double Robust}
Structured	{Imperative Searn, Dagger}
Understanding	{l1, audit, Prog. Validation}

1.4.1.1 Input Format

The input format for the VW is very intuitive. Actually, this is one of the reasons we chose to use VW. Not like most machine learning platforms, the VW system does not require from the user to create an index of all possible features and their counts in the data[7]. It seems VW was created especially for text analysis. The input data should list each data instance (record) on a separate line. To create a valid data input it is enough to start the line with the label or the class and a "|". Then list all the words in the data instance:

```
-1 | very simple data format
```

On the other hand, the input format also allows more complicated and useful formations. For example, each data instance can be assigned an importance weight:

```
1 0.5 | weighted data instance format
```

Another useful feature is the Namespace option. One can separate different kinds of features by grouping them under one Namespace. This option is very useful during

[6] The options used in this thesis are marked with bold.

[7] See for example http://www.cs.waikato.ac.nz/ml/weka/ for the *arff* data format.

feature selection. By using the *–ignore X* flag, one can choose to ignore all the features grouped under the *X* namespace.

```
1 0.5 |text weighted data instance format with namespaces |numeric 12 5 8797
```

The features themselves can be listed as in the examples above or one could specify the name of the feature and its value, using the equal sign:

```
-1 | length=5 count=3 country=Spain
```

And finally, it is possible to assign a weight per each feature in each data instance. For example to represent the sentence "to be or not to be" you can use the following format:

```
1 0.9 |details author=Shakespeare play=Hamlet |text to:2 be:2 or not
```

There are additional options, presented in the summary below:

Table 2. Vowpal Wabbit input format

Label [Importance] [Base] ['Tag] |Namespace Feature:float ... |Namespace:float Feature ... ... \n	
Namespace	String[:Float] Namespace is a mechanism for feature manipulation and grouping.
Feature	String[:Float]
Importance	a multiplier on learning rate, default 1
Base	a baseline prediction, default 0
Tag	an identifier for an example echoed on example output

1.4.1.2 Command line parameters and arguments

VW is a C++ library. The system is designed to be run from the command line. The manipulation of the machine learning algorithm and its parameters is done by inserting command line parameters and arguments. Table 3. lists all the parameters and arguments used in this work. To start the learning on the training dataset, the following format can be used to read training data from a file, create a cache and pass through the data n times. The resulting model is stored in the predictor.vw file:

vw -d train.txt -c --passes n -f predictor.vw

By adding the parameters from Table 3. and passing different values, we can change the way the machine learning model is trained.

After the training, the created model is tested on a different dataset. The predictions can be stored into the predictions.txt file:

vw -d dev.txt -t -i predictor.vw -p predictions.txt

In the Methods Chapter, we describe the specific commands used for our experiments.

Table 3. Vowpal Wabbit command line options

	Parameter	Use
1	[-d] [- data] <f>	Read examples from f
2	-- passes <n>	Iterate over dataset n times.
3	-c [- cache]	Creates a binary cache of dataset for faster loading next time (required with --passes n for n > 1)
4	-p [- predictions] <po>	File to dump predictions into
5	-f <filepath>	where to save final predictor
6	-t [- testonly]	Don't train, even if the label is there – used for testing
8	-ignore <a>	Remove a namespace and all features in it
9	--ngram <N>	Generate N-grams on features.
10	--skips <S>	Generate skips in n-grams. Can be used only in conjunction with n-gram parameter.
11	-power_t <p> [= 0.5]	Specifies the power on the learning rate decay.
12	-l [-learning_rate] <l> [= 10]	scales learning rate, default 0.5 or 10 for non-default rule
13	-loss_function {squared,logistic,hinge,quantile}	Switch loss function. Squared is default.
14	--l1 <value>	L1 Regularization default is 0
15	--l2 <value>	L2 Regularization default is 0

1.4.1.3 Vowpal Wabbit Algorithms

VW is an online learning system. Basically, it means that it learns as it goes. Let $X \in \mathbb{R}^{n \times d}$ be a dataset of n data instances with d features, $x_i \in \mathbb{R}^d$ be the ith data instance, and $y \in \{0,1\}^n$ be the corresponding set of binary labels, compute a prediction $\hat{y}_w(x) = \sum_i w_i x_i$ for every data instance. Next, compare the prediction with actual label y_i and update weights, so the prediction would be closer to true value. In order to update the feature weights Vowpal Wabbit uses Gradient Descent algorithm. Gradient Descent is a method for finding a minimum of a function by making step by step decisions about the direction you should head. As part of a machine learning algorithm, GD is minimizing a loss function. The algorithm's objective is to find the local minimum of the function. To find the best direction towards the minimum, we take the gradient of the function in a specific point and take one step to the opposite direction. The gradient is parallel to a derivative generalized to more than one dimension. Figure 4. illustrates graphically the intuition behind the Gradient Descent algorithm. We start at point x_1. Derivation of the function in a

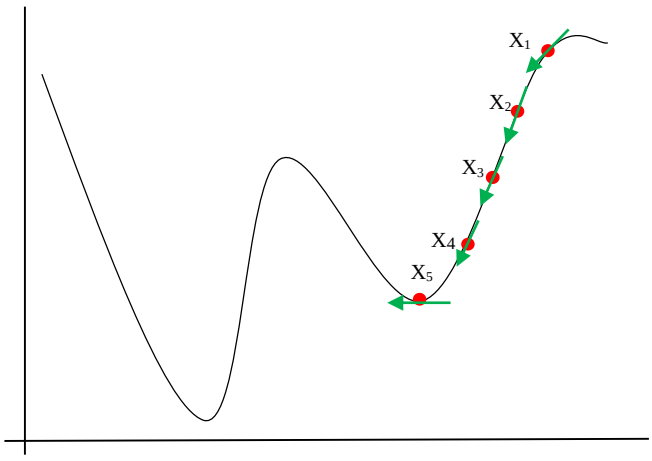


Figure 4 Gradient Descent

particular point gives us the gradient $\nabla l(w_i) = \frac{\partial l(\hat{y}_w(x),y)}{\partial w_i}$. The green arrow points in the direction of the negative gradient at points $x_{1...5}$. At each step, the following step is calculated with $w_{i+1} = w_i - \eta \nabla l(w_i)$. The derivation of the function gives us the slope of the line orthogonal to the function line at that point. At point x_5 the algorithm converges. At this point the slope equals 0, meaning that we reached the local minimum. Note that if we would start at a different point we could reach a lower minimum. At each iteration GD calculates the direction of the steepest descent with step size controlled by the learning rate, η.

$$\eta = \lambda d^k \left(\frac{t_0}{t_0 + w_t}\right)^p$$

Where w_t is the sum of the importance weights of all examples seen so far. With the available parameters, we can adjust the learning rate parameters λ using *-l* parameter and *p* with *--power_t* parameters. The learning rate is the size of the step we are taking. A higher learning rate will make the model converge faster – making larger step towards the minimum. However a very high learning rate may jump over the minimum, and the algorithm may not converge at all. One can adjust the *--power_t* parameter in the range [0,1]. 0 means the learning rate does not decay while 1 is very aggressive decay. Since we can run each iteration with a different step size, it makes sense to reduce the step size as we are coming closer to the minimum. This parameter allows us to start with a higher learning rate – making larger steps in the beginning, without being afraid to skip over the minimum[8].

Vowpal Wabbit optimizes per example the following function:

$$\sum_i \ell(x_i, y_i, w) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2} ||w||_2^2$$

Where λ_1 and λ_2 are the *L1* and *L2* regularization functions. $\ell(x_i, y_i, w)$ is the selected loss function. The parameters *--l1* and *--l2* specify the level (lambda values) of *L1* and *L2* regularization. *L1* is also known as Lasso regularization and *L2* is known as Ridge regularization. Ridge regularization parameter λ_2 controls the trade-off between fitting the training data and keeping the weights small. When $\lambda_2 = 0$ there is no regularization, and VW will optimize only the loss function. The highest the λ_2 parameter is, the prediction weights will become closer to zero. With Ridge

[8] https://github.com/JohnLangford/vowpal_wabbit/wiki

regularization all the features are included. Lasso regularization sets the weights for most features to zero, with a few set to a value larger than zero.

Given a prediction $\hat{y}_w(x)$ and a label *y*, a loss function $l(\hat{y}_w(x), y)$ measures the deviation of the predicted value from the true label. VW currently supports the following loss functions, with the Squared loss being the default.

Table 4. Loss functions in VW

Loss	**Function**
Squared	$\frac{1}{2}(\hat{y}_w(x) - y)^2$
Quantile	$\tau(y - \hat{y}_w(x))\mathrm{I}(y \geq \hat{y}_w(x)) + (1 - \tau)(\hat{y}_w(x) - y)\mathrm{I}(y \leq \hat{y}_w(x))$ Where I is the Indicator function and τ is the quantile.
Logistic	$\log(1 + \exp(-y\hat{y}_w(x)))$
Hinge	$\max(0, 1 - y\hat{y}_w(x))$

An online learning system updates the model as it goes through the examples. Such a system has a great advantage in speed. But it is important to be aware that it is also very sensitive to initiation and the order of the data. For example, if all the negative examples are in the beginning of the file, the system will perform poorly. It is important to randomize the data before learning.

1.4.2 Vector Space Model

We have used a Vector Space Model to support the implementation of the Snowball system (Salton, et al., 1975). Vector Space Model represents the target data instances and the seeds as vectors of attributes. Vector Space Model is most commonly used for Information Retrieval tasks. We want to represent the collection of our data in a document space, where D_i is the *i*th document in the collection. Each document is represented by an n-dimensional vector, consisting of n terms $D_i = (t_{i1}, t_{i2}, \ldots, t_{in})$. The terms can be weighted according to their importance. t_{ij} represents the weight of the *j*th term in document *i*. Given the vectors for two documents, it is possible to compute a similarity coefficient between them, $s(D_i, D_j)$. The similarity measure can be the inner product of the two vectors, or an inverse function of the angle between the vectors. When the document vectors are identical, the angle will be zero, producing a

maximum similarity measure. In Information Retrieval systems, we want to rank the documents according to their relevance given a query. Here we are using the same idea, only that our documents are the records – sometimes very short data instances and our query is the seed. We want to rank the data instances by their similarity to the seed vector. The vector space model method can be divided into three parts. The first part is indexing of the features from the data instances. We need to create an index containing all the possible features – to be able to use the same index across all data instances for the same feature. Then, create a feature vector for each data instance. The vector will have the count of the occurrences of each feature at the index associated with that feature. The result is a very sparse vector space, with many zero values. The second part is weighting of the vectors to improve the performance on the task at hand. The last part is ranking of the data instances with respect to the seed according to a similarity measure.

Before indexing, we need to decide how we want to engineer our features. In IR, it is common to disregard the stop-words (i.e. *the*, *a*) when indexing. Stop-words are believed to lower the performance of the retrieval because they are not informative with regards to the relevance of the document to the query. In the implementation of the Snowball system, we experimented with different stop-word lists to improve performance. Other linguistic features can be added to the index. For example, word n-grams, stemmed words, lemmas, part of speech tags or any combination of those. It is important to apply all the feature extraction techniques in all stages of the model building – the index has to contain all possible features; the seeds and data instances should be processed for feature extraction similarly in order to be comparable. Meaning, if the index does not contain POS tags, only word forms, the seed vector cannot contain POS tags. On the other hand, if the seed is stemmed and the data instance is not, the similarity score will be very low. In the Data Chapter, we review shortly all feature manipulation techniques used in this work. The Results Chapter describes the feature sets used in the experiments.

Term weighting has an influence on the precision and recall of the retrieval system. One wants to find as many relevant documents as possible without retrieving irrelevant documents. In the Snowball system, we want to extract new instances of named entities in the relation without selecting named entities of a different relation. The term weighting for the vector space model is applied based on single term statistics. We can refer to term weighting as a scheme of three types: local weighting, global weighting,

and length normalization. The scheme for term *i* in data instance *j* is represented by: $L_{ij}G_iN_j$. Local weighting ensures that terms that appear more often in the data instance represent this record better than terms that appear less often. Global weighting on the other hand takes into account the number of records in the dataset the term appears in. If the document frequency of the term is low, it means that this term is more specific to the data instance. For example, functional words will appear in all instances and will have a high local frequency, but they are less informative than content words. Length normalization is used for scaling vectors of different lengths. Long data instances will have more terms than short instances, which would make their chances to be retrieved higher. Cosine normalization reduces the impact of long documents. For local term weighting we have experimented with term-frequency weighting: $tf_{ij} = n_{ij}$ where n_{ij} is the row count of term *i* occurrences in data instance *j*; relative frequency: $rtf_{ij} = \frac{n_{ij}}{l_j}$ where n_{ij} is the raw count of term *i* occurrences in data instance *j* normalized for instance *j* length l_j; and log of the term-frequency: $logtf_{ij} = \log(tf_{ij}) + 1$. Additionally, following the work by Chisholm and Kolda (1999) we have used square root weighting: $iftf_{ij} > 0\ sqrttf_{ij} = \sqrt{tf_{ij} - 0.5} + 1$

Logarithmic weighing or square root weighting are preferable to raw frequencies. A score measured by raw frequencies would grow linearly as the frequency increases – meaning that if a term appears twice as many times in a record, the score with be twice as high. Logarithmic and square root functions are non-linear and give a better estimate of the instance scores. Global weighting was performed using IDF weighting – inverse document frequency $idf_i = \log(\frac{N}{df_i})$ where *N* is the number of data instances in the dataset, df_i is the number of data instances the term *i* appeared in.

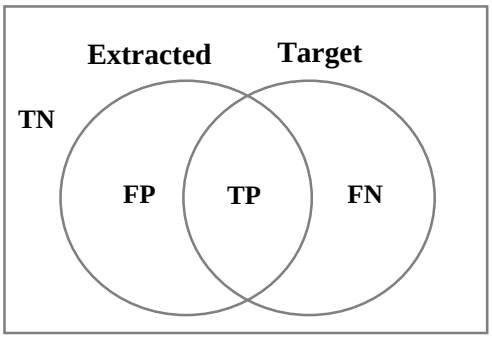
The similarity in vector space models is expressed via associative coefficients based on the inner product of the data instance vector and the seed vector, where feature overlap indicates similarity. The most popular similarity measure is the cosine coefficient, which measures the angle between the two vectors. Since we perform cosine vector normalization, the similarity measure is simply the dot product of the vectors.

1.5 Evaluation Methods

When talking about results of various tasks and experiments, one needs to keep in mind that results are relative to the difficulty of the task no matter how they are measured. For example, the accuracy of a morphological analyzer of English cannot be compared to an analyzer of Czech. Even though on the surface the task is similar, the performance on different languages cannot be expected to be the same. Unfortunately, for most NLP tasks there are no standardized evaluation sets for comparing systems. Without such a standard, how can we evaluate a new system?

In order to evaluate an information extraction system, we would need first to estimate an upper and lower bound for the expected performance. The upper bound is usually set by human performance on the task, namely, human annotation. Difficult tasks for humans are expected to be difficult for computers as well. The lower bound is the Baseline - the simplest algorithm with the simplest settings. A baseline can be based on frequency or co-occurrence. Our goal, when evaluating a system, is to gain improvement in performance over the baseline. In the best case, the system's performance will be reaching the upper bound. For the estimation of performance bounds and for evaluation of experiments we need to set the evaluation metric. The most common metrics for information extraction tasks are accuracy, precision, recall and the F-measure.

Going back to the definition of Relation Extraction task, we want to evaluate how well the system manages in extracting the right relations from the text instances without extracting wrong relations. We can measure how well our system performs using Precision and Recall. I will use the definition and illustration of Precision and Recall from Manning and Schütze (1999).



System	Target Relation	Wrong Relation
Extracted	TP	FP
Not Extracted	FN	TN

Figure 5. Illustration of Precision and Recall *(Manning & Schütze, 1999)*

Figure 5. illustrates the world of possibilities in extracting relations both in a graphical form and as a contingency matrix. After applying an algorithm for relation extraction, we have two sets: the set of the actual relations in the documents and the set of the relations we managed to extract. True Positives (TP) are the relations from the actual set that we managed to extract. False Positives (FP) are the relations we extracted that are not in the actual set, therefore, they are false. False Negatives (FN) are the relations we left behind – the relations from the actual set that were not extracted. True Negatives (TN) are the instances that do not contain any relation or containing a relation, not from the actual set, and our system did not extract those correctly. The numbers in the contingency matrix are the counts on instances from each region in the diagram. Precision is defined as a measure of the proportion of extracted items that the system got right. In other words, out of everything the system extracted, how many are correct:

$$precision = \frac{TP}{TP + FP}$$

Recall is defined as the proportion of the actual target instances that the system extracted. Or, how many relations were extracted out of the actual set:

$$recall = \frac{TP}{TP + FN}$$

In a relation extraction system, it is possible to trade off Precision and Recall. If we "extract" the target relation from all the text instances, we will be able to gain 100% recall. All the target relations will be obviously extracted but with them, all the false positives as well, dragging precision down. Essentially, when building an extraction system, we need to decide if we are striving for high precision, not allowing false positives in our outputs, or high recall, where we would like to see more results even if some are not correct. There exists another measure that combines precision and recall scores into one measure of performance –the F-measure. In the definition bellow, we assume equal weight for precision and recall, although different weights are possible.

$$F = \frac{2PR}{R + P}$$

Where P is precision and R is recall.

An additional common measure of performance is accuracy. Accuracy is simply the percentage of correctly labeled entities, either as correctly extracted or as true negatives. Manning and Schütze (1999) list in their book several disadvantages of

accuracy over precision and recall. In the type of tasks, we are dealing with the number of true negatives is significantly larger than the number of entities in all the other sets. The accuracy calculation is not sensitive to this disproportion while recall and precision are. When all set counts are equal, the F-measure will produce higher scores when extracting true positives, where accuracy will only count with the number of errors. With giving different weights to precision and recall, we can create a more flexible measure, which would correspond to our preferences.

2 Data

2.1 Description and Analysis

The data used in this thesis was provided by an industrial partner. The corpus contains real customer data, describing personal information and points of interaction with the company. The database includes structured and unstructured information that are related to a specific party. The data given for the purpose of the thesis contains data from 2006 to 2015. As this data is private, it is highly confidential. The data cannot be shared, copied or described with details that include names and other identification information. We have tried to provide the most informative data description as possible, given this limitation.

For the experiments, we used information from two main tables. The *Party* table consists of structured information about business parties. The other table - *Text* - was the source of the unstructured data. The Text table contains a description of one interaction point of the parties with the company. The descriptions are between 1 to 1,631 tokens long, spanning over 1 to 10,000 characters. The number of records in the Text table is 1,931,651. To establish the data to be used in the thesis, the tables were joined into one table that would consist of textual unstructured information associated with structured data about the related party. Since our relation involves a Person, the joined table was split by the type of the party into persons and organizations. In the final *Person* table, there are 1,202,845 records corresponding to the number of people who had a text description associated with them. As we are interested in recognizing the origin country, we extracted the information related to foreign clients.

The same party entity may have more than one interaction point with the company. Moreover, the description length varies from one word to lengthy texts. For the binary *Country of Origin* relation, we took as data instances the occurrences of the named entities of interest with the context surrounding them within a predefined window. Meaning, that many of the text descriptions were not considered at all. On the other hand, some text descriptions yielded several data instances. The corpus was then split into three sets: training, development and test. Figure 7. illustrates the preparation of the datasets.

Unfortunately, the vast majority of the textual data could not be used for the demonstration of the methods implemented here. The data to be used was constrained in two ways. Firstly, the text has to contain a named entity. Secondly, the same attribute

expressed by the named entity has to be part of the structured data, to be used in the Distant Supervision implementation and the Snowball evaluation. The only attribute we found having both conditions was the country of origin. We had to exclude Czech clients from the dataset since the mentions of Czech Republic in the text had many different senses, compared to the mentions of the foreign countries.

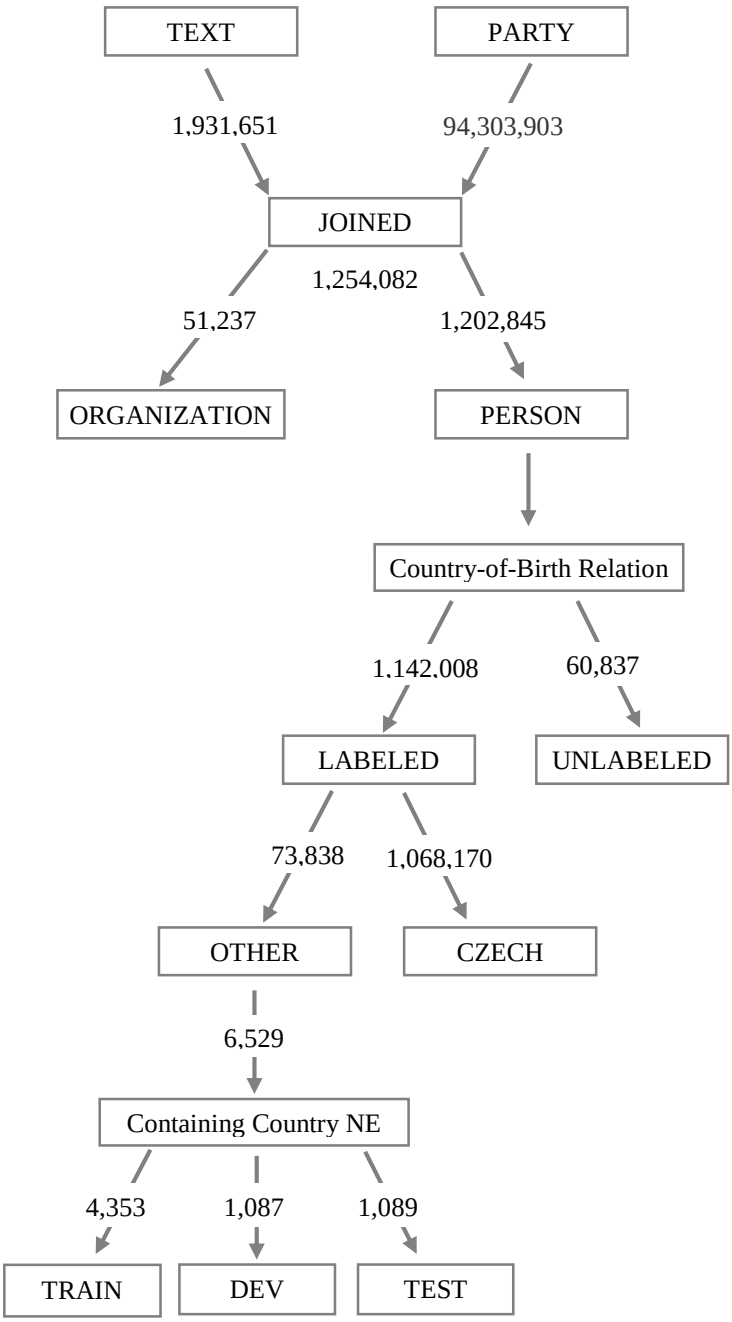


Figure 6. Dataset Preparation

The datasets were created the way that each dataset contains the same distribution of classes – 72% negative examples to 28% positive examples. For the Distant

Supervision method, we have used the training data set to train the models, the development dataset to select features and tune the parameters and the test dataset to test the final model. In Snowball, we combined the training and developments datasets to tune the system's parameters and select features. The test dataset was used to test the tuned system.

Table 5. Distribution of classes

	TRAIN	DEV	TEST
Positive examples	1,234	309	309
Negative example	3,119	778	780

The data domain presented two main areas of challenges. The first one originates from the structure of the database and the way the structured information is stored in the database. The structured data consists of keys and flags. The details of a person are then presented as an array of numbers, i.e. [34, 45, 5]. In order to translate the keys into their meaning, one needs to access a translation table, where the keys are mapped. The translated description is more accessible – [Czech Republic, Prague, Married]. The mapping of the keys to their meaning is most important in relation to the Distant Supervision method. When extracting the *country-of-origin* relation, we compare the structured *country-of-origin* column to the countries found in the text. The column contains numbers, where each number signifies a country. In order to compare the values from the column to the values extracted from the text, they have to be translated first. Additionally, some of the structured data is filled in incorrectly or in a non-unified fashion. Comparing the column containing *country-of-origin* key with another column containing *place-of-birth* description, we noticed that some of the data was filled in incorrectly. Namely, the description of the place of birth did not correspond to the country key in the next column. That means that either there is an error in the country key or in the description. Moreover, the place of birth description contains a mixture of cities and countries, with many different entered values for the same entity, for example, {SR, Slovensko, Slovenska Republika}, not to mention spelling mistakes and typos. This is surprising because such data is usually referred to as structured or at least semi-structured and we did not expect such irregularities in this part of the data. Those issues pose a serious problem for our methods. The unstructured data in not manually labeled, and we are using the structured data to supervise the learning. If there are errors in the data it will affect the performance of the algorithms. In order to

estimate the percentage of badly entered data, we manually counted how many mismatches and mistakes appeared in the *country-of-origin* and *place-of-birth* columns in the *Foreign* table. Out of 100 records, 2 were labeled with a non-matching country code, 4 birth place entries were misspelled and 10 were invalid values, making it 16% of invalid entries. The majority of mistakes are made in countries other than Slovakia – constituting about 50% of the *Foreign* table.

The second challenge is in the unstructured part of the database. The unstructured data contains texts that were submitted by company's personnel, as a transcript of the interaction point with the client. Since the description is for internal purposes only, the text contains many company specific abbreviations. It seems the writers of the descriptions were not concerned with the correctness of the texts. We encountered spelling mistakes and missing white space after punctuation. In the next section, we address the challenges faced when processing the data on different levels of analysis.

2.2 Data Preparation

In this section, we discuss the modification applied to unstructured data and the additional analysis performed on the tokens, such as Lemmatization, Stemming, and named entity recognition.

2.2.1 Text Normalization

2.2.1.1 Punctuation

In order to minimize the errors caused by wrong spacing after punctuation, we added a space after punctuation if it was preceded by a string of letters and followed by a string of letters, excluding @ from the punctuation list to avoid parsing emails.

In some of our experiments, we discarded punctuation and numeric characters entirely when building the feature vectors. In others, we replaced all punctuation by "PUN" string to group all possible punctuation strings into one. The same procedure was applied to the numeric characters replacing them with "NUM" string. The reasoning for discarding or replacing non-letter characters is the possibility that they introduce noise without adding information for our classification.

2.2.1.2 Capitalization

Many natural language processing application can benefit from normalizing the text by lowercasing the tokens. We want to consider tokens which are uppercased in the beginning of sentences as the same feature as the same token in the middle of the

sentence. On the other hand, some capitalized words are uppercased for a different reason – names of people, organizations and location are capitalized as well. This information is essential for recognizing named entities and differentiating names from regular words, for example, Apple and apple. In this work, we decided to lowercase tokens after applying the named entity recognizer. In this way, we first provide complete information to the NER allowing it to use capitalization as a feature. Then we generalize over capitalized and lowercased tokens in the rest of the analysis.

2.2.1.3 Stop-words

Stop-words are functional words such as prepositions, determiners, and conjunctions. In many applications, such words are removed from the text while processing because they are very frequent and uninformative. When we classify text instances we want the features to be selective for the class. For example, the word *capital* is probably very important to identify *is-capital* relation, but the word *the* would appear in most of the sentences, not giving us a hint of which class the sentence belongs to. On the other hand, some prepositions are very important in the task of relation extraction. A proposition is sometimes the whole context between named entities, as in *Bob from England*. In those cases, stop-words should not be ignored. We decided to put the stop-words elimination to the test and created experiments with and without stop-words. We used a compiled list of 256 stop-words for Czech[9]. We have also experimented with different stop-word lists. The second stop-word list is a subset of the complete list, excluding prepositions that could be beneficial for out relation extraction methods. We have also compiled a stop-word list from company's abbreviations.

2.2.2 Stemming

Stemming refers to the removal of characters at the end of the word in an attempt to grasp the core of the word. Most algorithms for stemming are built to remove suffixes leaving only the stem of the word. For example, with stemming we can deduct that *writer* and *write* share the same root, they are semantically related. Note that both *writer* and *write* are in their base dictionary form. The main idea behind stemming is similar to the reasoning for lowercasing tokens. We want to be able to generalize over variant forms of a word because the variants originate from the same semantic

[9] https://sites.google.com/site/kevinbouge/stopwords-lists

meaning. As with other normalizing techniques, stemming may improve the results of one task on a specific collection but may reduce the performance over a different collection or a different task. We have experimented with both stemmed features and non-stemmed features.

Stemming was performed by the Snowball[10] stemmer for Czech that was ported to python from the Java implementation by Ljiljana Dolamic, University of Neuchatel[11] (Porter, 2001). The stemmer is implemented in two modes: light and aggressive. The light stemmer removes grammatical case endings from nouns and adjectives, possessive adjective endings from names, and takes care of palatalization. The aggressive stemmer also removes diminutive, augmentative, and comparative suffixes and derivational suffixes from nouns. In our experiments, we used the aggressive stemmer.

2.2.3 Lemmatization

Lemmatization is similar to stemming, but instead of chopping off letters from the end of the word, it reduces the word to its base form or dictionary form. Taking the examples *write* and *writer*, both words are in base form, therefore, would not be changed by lemmatization. On the other hand, *writers* and *writes* would be reduced to *writer* and *write* correspondently. In a way, lemmatization reduces inflectional morphology and stemming reduces derivational morphology. In this thesis, we used a morphological analyzer – MorphoDiTa (Straková, et al., 2014). MorphoDiTa stands for Morphological Dictionary and Tagger. It performs full morphological analysis, including lemmatization, part of speech tagging and tokenization. The analyzer comes with a trained model for Czech, which achieves state-of-the-art results. It is possible to train the analyzer for other languages using annotated data. MorphDiTa was developed specifically to cope with morphologically rich languages, such as Czech. As other Slavic languages, Czech is an inflective language. Most of the inflective morphology of Czech is expressed through suffixes, marking cases, gender, and number. In lemmatization algorithm, all of those suffixes have to be identified and removed to get to the base form. MorphoDiTa is easily suited for different inflective languages because it is not using any language-specific knowledge. Instead, it creates

[10] The Snowball Stemmer, not to be confused with the Snowball system for RE.

[11] http://research.variancia.com/czech_stemmer/

templates it deducts from clustering common form endings. Based on an annotated dataset, the system collects all word forms labeled with the same lemma and builds a trie structure. Since in Czech inflective morphology is expressed via suffixes, word forms of the same lemma share the same prefix. In a trie, the prefix is expressed with linked nodes, one node per letter. The suffixes are then branch out of the end of the prefix as shown in Figure 8. (Straková, et al., 2014). When looking for a suitable prefix, the system takes into consideration the length of the suggested suffix, not to exceed a predefined length. Namely, the suffix cannot be too long. On the other hand, the system tries to find the longest prefix. The template is set of all the suffixes of the defined prefix. New lemmas either share the template defined by a different lemma or create a new template. Our algorithm is using MorphoDiTa Python bindings. We input raw text and the system outputs the corresponding lemma and part of speech tag per word form.

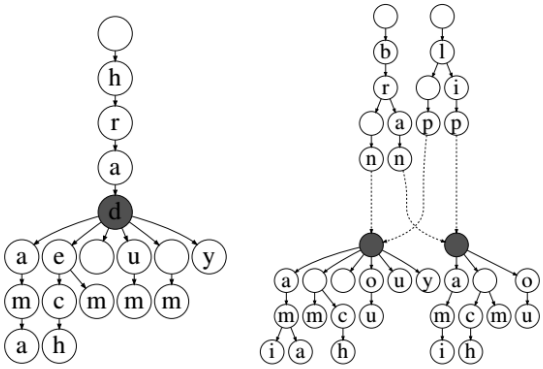


Figure 7. Trie representation of word forms in MorphoDiTa. *Taken from Straková, et al (2014)*

2.2.4 Part of Speech Tagging

Part of Speech (POS) Tagging refers to the assignment of a grammatical tag such as noun, verb or adjective to a word. As for lemmatization, we used the MorphoDiTa tool to POS tag the data (Straková, et al., 2014). The POS tagger of the system is going hand in hand with the lemmatization algorithm. After the system suggests a list of possible lemma – tag pairs, the tagger performs disambiguation for choosing the most probable tag in the context. The tagger is implemented as supervised, rich feature averaged perceptron. MorphoDiTa used the Prague Dependency Treebank 2.5 (Bejček, et al., 2012) for training. In our algorithm, we used the 2-position tags provided by the system corresponding to POS in the first position and sub-POS in the

second. Figure 9. presents the distribution of first position POS tags in the data. It is surprising that only 19 tokens were not identified (labeled X), given that many tokens are domain specific abbreviations.

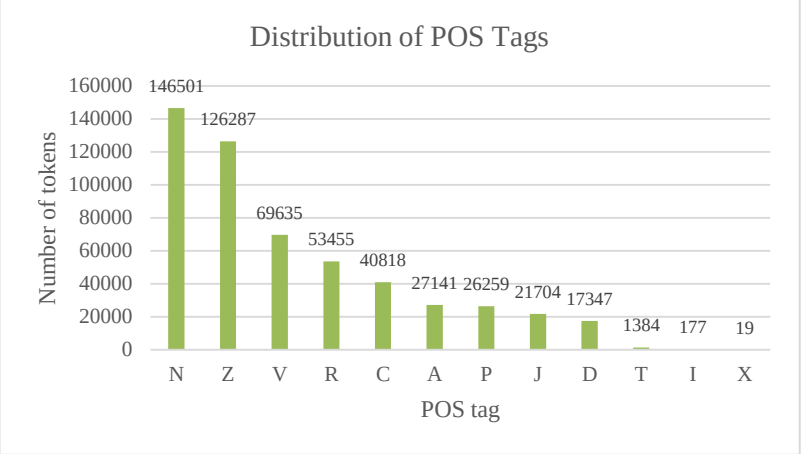


Figure 8. Distribution of POS Tags

Table 6. POS tags description

Value	Description	Value	Description
A	Adjective	P	Pronoun
C	Numeral	V	Verb
D	Adverb	R	Preposition
I	Interjection	T	Particle
J	Conjunction	X	Unknown, Not Determined, Unclassifiable
N	Noun	Z	Punctuation (also used for the Sentence Boundary token)

2.2.5 MorphoDiTa Evaluation

The MorphoDiTa system we used for POS tagging and lemmatization was trained on normalized manually annotated texts. POS tagger was trained on PDT 2.5 which contains a collection of Czech newspaper texts (Bejček, et al., 2012). We consider here newspaper texts as normalized, because in comparison to the texts in our collection we do not expect so many errors, misspellings, and spacing issues. The same can be claimed with regards to the lemmatization training data. In order to evaluate how well MorphoDiTa copes with our irregular data, the results on a small section of the data were manually evaluated. To simplify the task, only the first position of the POS tags was considered for evaluation. Table 7. summarizes the results of our manual evaluation on 100 data instances, on both lemmatization and POS tagging. We also brought the results reported by Straková et al. (2014) for comparison.

Table 7. MorphoDiTa performance

	Accuracy on Business Domain	Accuracy reported by Straková et al. (2014)
Lemma	88.5%	97.80%
POS Tag	96%	99.18%

This evaluation was performed for estimation only. The amount of manually annotated instances is not enough for a definitive evaluation. That said, we have noticed interesting domain specific errors. Lemmatization process is expectedly failing on misspelled words, while the POS tagger is guessing the tag correctly in many cases, even of unrecognized words. A difficult lemmatization task is to recognize domain specific abbreviations. But if such abbreviation is in the dictionary, lemmatization can help with text normalization, as in the case of the lemma *číslo*, which appears in the text as *č*. In Lemmatization, we encountered a small number of errors that are not domain specific, amounting to 2%. This percentage corresponds to the error rate evaluated by Straková et al. (2014). We expect that the performance would be worse if we did not fix the spacing issue before the morphological analysis.

2.2.6 Named entity recognition

One of the most important components of RE is Named Entity Recognition (NER). Named Entities constitute the building blocks of relations. The system we used for NER is the NameTag (Straková, et al., 2013). It is based on a Maximum Entropy Markov Model, with a Viterbi algorithm which is used to decode probabilities estimated by the maximum entropy classifier. NameTag achieves a state of the art results for Czech with 82.82% F-score on the Czech Named Entity Corpus. As with MorphoDiTa, the recognizer is available with a trained language model for Czech. The NER is classifying named entities into a two-level hierarchy. There are 7 upper-level classes, including numbers, time expressions, personal names, geographical names, and institutions. The lower level consists of fine-grained 42 named entity classes (the full hierarchy can be found in Attachment 1 (Ševčíková, et al., 2007)). In this work, we only use the Country named entities. We manually evaluated NameTag on the Country label. The results are presented in Table 8. The results reported by Straková et al. (2014) are brought only for orientation. Those scores were calculated for all of the classes in the hierarchy while our evaluation considered only one upper-level and one fine-grained class.

Table 8. NameTag Evaluation

	Upper Level			Fine-Grained		
	Recall	Precision	F-score	Recall	Precision	F-score
Reported By Straková et al. (2014)	-	-	80.30	-	-	77.22
Manual Evaluation	86.11	68.38	76.23	81.48	72.13	76.52

For evaluation we manually labeled 100 data instances with the class GC, corresponding to Countries and States. We then labeled the data instances with NameTag and compared the results. We counted the number of correctly labeled entities with the label GC and entities labeled with another geographical tag. Most of the countries were correctly labeled with GC, confirming that an upper-level geographical label would not provide better performing models. That is, taking a high-level tag would result in unnecessary noise as demonstrated in lower precision in the manual evaluation. Most of the errors resulted from incorrect labeling of domain specific abbreviations, which are considered by the tool as countries. In Table 7. we present the most frequent GC labeled tokens. Table 7. covers over 75% of the extracted NE. The tokens labeled “PK”, “KC”, “IT”, “CZK” are not countries. The label “PK” was assigned to 13.13% of the extracted NE. The reason for such errors is the fact that the same abbreviation can signify a country. For example, PK is the ISO code that refers to Pakistan and IT is the ISO code of Italy. Some of the countries were not caught by NameTag because of parsing errors, where white space was missing between words. Some countries were labeled with GU, a class of Castles. Considering that most of GU labeled entities are actually cities, we decided to include only the GC tag in our models.

Table 9. Most frequent Country labels

NE Label	Count	Presentage	NE Label	Count	Presentage
ČR	1,040	22.28%	České	62	1.33%
PK	613	13.13%	Slovenské	60	1.29%
Slovensko	414	8.87%	Německu	56	1.20%
Slovensku	249	5.33%	IT	55	1.18%
KC	186	3.98%	Polsku	52	1.11%
USA	130	2.78%	Německa	41	0.88%
Slovenska	96	2.06%	Slovenská	40	0.86%
SR	87	1.86%	Ruska	37	0.79%
Ukrajině	81	1.74%	Rusku	31	0.66%
Ukrajinu	74	1.59%	CZK	28	0.60%
Polska	64	1.37%	Česká	25	0.54%

3 Methods

In the original implementations of the Snowball system and the Distant Supervision method as presented in this work, both rely on finding all the named entities of interest in the text. The methods are using the context words surrounding the named entities as features. It is important to note that in the data used here, only one of the named entities is present in the text. In this work, we use the structured data entries as the second named entity in the relation. We rely on the fact that a specific text instance is related to a party, given to us from the structure of the database. More specifically, the *country-of-origin* relation is a relation between two entities <Person, Location>. The name of the person is part of the structured data. The name is rarely present in the text. Moreover, the person's name and the location of their origin are not mentioned in any of the data instances together. But, we know that each text instance is related to one person – the client. In our implementation, we take the context surrounding only one of the named entities in the relation – the country.

There is another important difference between the traditional relation extraction and the methods presented in this thesis. Similarly to the approach of Craven and Kumlien (1999), we do not extract pairs of named entities in the given relation directly. Instead, we classify the context as informative for the given relation or not informative, making the task a binary classification task. In other words, if the context surrounding the country expresses the *country-of-origin* relation, we are assuming that the mentioned country is the birth country of the client. After classifying the data instances, we can pair the named entity from the text to the name of the client from the structured data to provide a list of pairs. Take, for example, the following sentences:

1. *The client is a citizen of Canada.*
2. *After a vacation in Spain, the client opened this request.*

The context in the first sentence is informative. Reading the sentence we can tell with some assurance that the client related to this text is from Canada. Compared, the second sentence does not provide us with similar information. Our methods are built to detect this difference.

Changing the task to a binary classification task required relabeling of the data. In the implementation of the Distant Supervision method by Mintz et al. (2009), the structured data was used as labels. In our data, the structured data we could use in such a way is the *country-of-origin* column in the *Party* table. In order to create binary labels

to train the Distant Supervision models and for the evaluation of the Snowball system we constructed a simplifying assumption: If the country mentioned in the text is the same as in the structured data, we consider this data instance a positive example. The rest are negative examples. This assumption greatly simplifies the task at hand. There are over 150 different countries in the database. We have 6500 data instances. This is too small of a dataset to classify the data to 150 labels. On the other hand, we are aware that this assumption brings noise with it. It is not necessary that the context in which one would mention their country of origin would actually be informative for this relation. To summarize the above modifications:

1. The textual data contains only one of the NE from the relation.
2. We consider the task as a binary classification of the NE context.
3. We assume that if a country mentioned in the text is the same as in the structured data, the context expresses the relation.

In the following sections, we will describe the dataset preparation and the implemented methods in length, with the three modifications described above in mind.

3.1 Data preparation

The data used here is hosted on a cluster, with many tables containing different kinds of information. The description of the tables resided in a different system. For the preparation of the datasets, we used Hive - data warehouse software for reading, writing, and managing large datasets residing in a distributed storage[12]. All the data related procedures were run from the company's server using SQL-like Hive commands. The prepared data was fed as input to Python scripts for further analysis and processing.

The Snowball algorithm requires the data instances to be converted to feature vectors. The Distant Supervision method has a different data representation following the format of Vowpal Wabbit. Nevertheless, in both methods, we need to extract features to represent the data instances. The features used in the original methods are constructed from the context words between the named entities to the left side of the first entity, and to the right of the second. As we do not have both named entities in the text, the datasets are prepared with features only from the left and right side of the NE. Actually, the NE in the text is the second NE in the relation, meaning that our

[12] https://hive.apache.org/

context is the right context and the middle and left merged together to the left context. As was mentioned in the previous chapter the named entity recognition was performed using the NameTag tool (Straková, et al., 2013). We included only the data instances that contained named entities tagged with the label GC. We created various datasets with several options for the number of tokens taken from each side of the named entity. Another variation in the datasets is in the processing applied on the extracted tokens. For each method, we engineered and evaluated different feature set, optimized for the performance of the individual method. We have prepared the data to be used in the experiments using the following scheme:

Table 10. Data preparation for country-of-origin relation

1	**For every record:**
2	Extract named entities
3	**For** every named entity:
4	**If** the named entity is a country:
5	Extract the surrounding words around the NE
6	**Process** the extracted words to add linguistic features:
7	Normalization
8	Lemmatization
9	Stemming
10	POS tagging
11	**If** the country in the text is the same as in the country-of-origin column:
12	Label the record with 1
13	**Else**:
14	Label the record with -1
15	**Write** the processed tokens as features into the output file
16	**Split** the data into Training, Development and Test datasets

As the baseline dataset, we extracted only 1 context token from each side of the NE. We did not apply any normalization procedure. Other datasets included lemmas and POS tags produced by the MorphoDiTa (Straková, et al., 2014). We have also used a stemmer to include stems in some of the datasets. Additionally, we experimented with n-grams up to a third degree. For the Distant Supervision method, all the produced datasets were split into Training, Development and Test datasets. We have kept the same distribution of classes in all three datasets. The training dataset was used to train the machine learning methods with different parameters and features. The

development dataset was used to test the models while tuning parameters and selecting features. The test dataset was used to test the final model with the selected features. The Snowball System does not involve training, therefore, we combined the training and development datasets for tuning the system parameters and for feature selection. The test dataset was used to test the performance of the system.

3.2 Feature Engineering

One of the most influential parts of any machine learning or bootstrapping algorithm is the selection of features. Engineering good features suitable for the selected method is sometimes a challenging task. Our goal is to find the best performing set, with the least computational effort spent on creating the features. All of the features used in our implementation are based on the words surrounding the named entity in the data instance in which they appear. We experimented with several types of features: word forms, part-of-speech tags, lemmas, stems, in different context windows. We have evaluated different feature sets separately for the Snowball system and for the Distant Supervision method. The best combination of features was chosen in the following way: First, we evaluated the effect of window size on word form features. The size of the context surrounding the words affects the amount of information the system is receiving. One might think that the largest the context window the better are the results. But this is not the case. Especially when the model used is a bag of words, we might confuse the system with irrelevant features. Each task has a different ideal ratio between giving enough of information without giving irrelevant one. In our task, we have found that a rather small window is required since the words closest to the named entity are the most informative. Taking the best-performing window we tried to improve performance by removing stop-words. Stop-words are frequently used words such as functional words (e.g. determiners and prepositions). Some algorithms can benefit from stop-words removal because usually those words are less informative. We have used three different stop-word lists. The first list is constructed from a complete list of 256 stop-words for Czech[13]. The second stop-word list is a subset of the complete list, excluding prepositions and adverbs that could be beneficial for the relation extraction methods. We excluded the prepositions {*do*, *na*} used with locations. We also excluded adverbials which refer to locations such as {*kde*, *daleko*}.

[13] https://sites.google.com/site/kevinbouge/stopwords-lists

The reduced list contains 239 stop-words. The third list contains company specific abbreviations. The third list was also used in combination with the other two stop-word lists. In the following experiments, we exchanged numeric values and punctuation by PUN and NUM respectively, following by experiments ignoring those value entirely. Punctuation rarely holds additional information with regards to the relation we are extracting. We assume that the methods would benefit from the removal of punctuation tokens. Numerical values are even less informative. The data contains a lot of different numerical tokens, making similar feature vectors seem farther away than they really are. Taking the best performing setting up till now, we changed word forms to lemmas and stems. Lemmas and stems contribute to the generalization of the models. By converting word forms to lemmas or stems we collapse several word forms into one. Since the semantic core meaning of the word is usually preserved when it is lemmatized we should gain from such a generalization. Take for example the following sentences:

1. *Klientka se vrací na Slovensko.*
2. *Klient se vrátil na Slovensko.*

The meaning of the sentences is very similar, with identical informatively with regards to the relation. With word forms, the two sentences are represented with two different feature vectors. After lemmatization, the two sentences would have identical feature vectors: {*Klient se vrátit na Slovensko .*}. In the subsequent experiment, we added POS tags as well.

In an attempt to improve the results further, we added bigram and trigram features. N-grams are useful for a more accurate representation of the context. Some words on their own have a different meaning in a sequence with other words. Taking the example above, we would rather consider *se* as one feature together with the verb. Since other occurrences of *se* can be less informative in a combination with other verbs. Vowpal Wabbit offers an n-gram parameter, which creates n-grams with different n value, and uses those during training. Another useful parameter is the skip-grams which can be used in combination with the n-gram parameter to create n-grams by skipping over a predefined window of words, making the features in the form of *n-skip-k-gram*. Those features are useful for detecting distant relations between words.

3.3 Snowball

The Snowball method described in the Background chapter counts with having multiple appearances of each pair of related named entities in the same context. It is clear that this assumption of having many examples of both NEs in the same data instance is not valid for this domain data. Not only that the pairs of NEs are not repetitive, but the data instances contain only one NE, having the other as part of the structured data. Given those differences, we had to modify the algorithm. We describe the original Snowball implementation below.

Table 11. The Snowball system by (Agichtein & Gravano, 2000)

1	**Find** seed NE pairs in the text
2	**For** every pair:
3	Extract the words around the NE pair
4	Prepare LP vector from the words to the left of NE1
5	Prepare RP vector from the words to the right of NE2
6	Prepare MP vector from the words between NE1 and NE2
7	Normalize and scale the vectors by a factor of relative importance
8	Prepare pattern from LP, RP, MP vectors
9	Extract named entities from the text
10	Find sentences with NE pairs of the same type as in the relation
11	Convert sentences to pattern format using steps 3-8 – Create and normalize LT, RT, MT
12	**For** every seed pattern:
13	**For** every text pattern:
14	**If** both NEs are of the same type as in seed pattern:
15	Calculate Match Score ($Match(P,T) = LP \cdot LT + RP \cdot RT + MP \cdot MT$)
16	**Else:** Score = 0
17	**Rank** scores
18	**Extract** NE pairs from text patterns according to a similarity threshold
19	**Evaluate** extracted NE pairs
20	**Collapse** patterns vectors to their centroid, use those vectors as the seed patterns
21	**Repeat** from line with newly extracted NE pairs

In our implementation of the Snowball, there are several differences from the original implementation above. We could not have seeded the process with pairs of named entities. Our text instances do not contain both named entities of the relation. Moreover, the NE that appears in the text might be part of a different relation, such as the location of a vacation or the location of a workplace. Seeding with the contexts

surrounding any Country would bring noise to the system. It would have been impossible to identify which of the context vectors express the relation *country-of-origin* and which express a different relation. Therefore, we have decided to seed the system with context words directly. We have manually constructed two seed vectors of possible features to serve as the seed pattern for the right and left context. The seed features were selected in a trial and error method. The initiation of the system had the largest impact on the results of the extraction. If we seed the system with too many options, the algorithm extracts many data instances at the first iteration and fails to extract any relevant data instances at the next iteration. As in the Snowball system by Agichtein and Gravano (2000), we only consider sentences that contain the Country labeled named entities. The data instances were labeled with named entities during data preparation. We extract the context words in a predefined window to the right and left of the labeled NE. We have changed the pre-processing of the words in the feature engineering stage. We have first experimented with word tokens. We then added different normalization procedures – stop-words removal, removal and normalization of punctuation and numbers. Stems, lemmas and part of speech tags were added to some of the datasets. Finally, the features were converted to the vector format. After the vectors were constructed, we applied different weighting schemes. We have used cosine length normalization for all of the vectors in the system. For term weighting, we experimented with term frequencies, relative term frequencies, Log normalization and SQRT normalization. For global weighting, we used IDF.

At that point, we have two seed vectors – LS and RS, corresponding the left seed pattern and the right seed pattern respectively. We also have a collection of data instances all converted to feature vectors. Each data instance is now presented as two vectors one for each side of the named entity. As in the original Snowball system, we take the dot product of the corresponding context vectors and sum them up to calculate the similarity score between the seed vectors and the data instances:

$$Match(S,T) = LS \cdot LT + RS \cdot RT$$

In order to identify which instances are most similar to the seed pattern, we rank the Match scores. The best ranking data instances are processed in two ways. First, the data instance is labeled as 1 – meaning considered a positive example. It also means that we can extract the named entity from the data instance and the second named entity from the structured data to produce an NE pair in the relation. Then the features from the data instance are evaluated. The best features are added to the seed pattern

for the next iteration. To choose the best ranking data instances we have added a similarity threshold as a parameter. During parameter tuning, we experimented with different thresholds to produce the best trade-off between recall and precision. If we set the similarity threshold parameter too low, we allow data instances that are barely similar to the seed pattern to be extracted. This way we increase recall, with many instances passing through the bottleneck. But at the same time, we are decreasing precision, labeling data instances that are not similar to the seed as positive examples. Another parameter in the system is connected to feature evaluation. After we have extracted the similar data instances, we want to enrich the seed patterns with their features. We could add all features to the seed. But, this would add unnecessary noise to the seed feature vectors. Instead, we use a count of overlap of each feature in all extracted data instances in a given iteration. The intuition behind this decision is that features that appear in more extracted data instances are more likely to be connected to the relation we are trying to detect. We add to the seed vector only features that are not part of the seed already. The overlap count threshold parameter was tuned in our experiments. Here, similarly to the similarity score threshold, we consider recall and precision trade-off. If we set the count too high, there might be not enough features above this count to enrich the seed vectors. Without enrichment, the next iteration will not extract additional data instances. The recall and precision, in this case, will stay the same as in the iteration beforehand. If we set the overlap count threshold too low, the seed vector will be enriched with uninformative features. The recall, in this case, would increase and the precision would decrease.

According to Agichtein & Gravano (2000), the middle context bares the most informative features. Therefore, in their implementation, the middle vector is scaled to have higher weights. In our implementation, the left vector is comparable to the middle vector. We have experimented with seeding only the left patterns to examine their claim.

The whole system is implemented in Python. We have used external libraries such as *ufal.morphodita* and *ufal.nametag* for morphological analysis and named entities recognition. Stemming was performing using a ready to use implementation of the Snowball stemmer for Czech[14]. All other components of the system were implemented using the standard libraries.

[14] http://research.variancia.com/czech_stemmer/

The following description depicts the algorithm of our system:

Table 12. Modified Snowball Implementation

1	**Tag** the data with **Named Entities**
2	Select the data instances containing GC NE label
3	Construct **seed patterns**
4	Convert seed patterns into two feature vectors – Lseed, Rseed
5	Weight the seed vectors with one of {raw,relative,sqrt,log} local weights
6	Weight the seed vectors with IDF global weight (optional)
7	Normalize and scale the seed vectors
8	**For** every data instance:
9	**If** data instance was not labeled already:
10	Extract the words around the NE
11	**Process** the words for linguistic features {stem,lemma,POS-tag}
12	Prepare LT vector from the words to the left of the NE
13	Prepare RT vector from the words to the right of the NE
14	Weight the text vectors with one of {raw,relative,sqrt,log} local weights
15	Weight the text vectors with IDF global weight (optional)
16	Normalize and scale the text vectors
17	Calculate **Match Score** ($Match(S,T) = LSeed \cdot LT + RSeed \cdot RT$)
18	**Rank** scores
19	**If** score above **similarity** threshold:
20	**Label** the data instance as positive example
21	**Collapse** extracted vectors into Left and right vectors
22	**For** every feature in the collapsed vectors:
23	**If** feature count is above **overlap** threshold:
24	**If** feature not in the corresponding seed vector already:
25	**Add** feature to the corresponding seed vector
26	**Repeat** from line 5 with the new seed vectors

3.4 Distant Supervision

Following the work of Craven and Kumlien (1999) and Mintz et al. (2009), we implemented the Distant Supervision method that exploits the structured part of the database for training data. We used values from the Country column as weak labels for the text instances. As mentioned before, we consider every data instance that contains the named entity identical to the value in the Country column a positive training example. All other data instances, which contain named entities tagged with

GC (Country) other than the one in the structure data, are considered negative examples. We then trained a classifier using the weakly labeled data. The classifier is trained to identify whether the context words express the relation of interest. For all the models implemented as part of the Distant Supervision method, we have used Vowpal Wabbit. We have used a simple unigram model as a baseline, with word tokens from the surrounding context of NE represented as a bag of words. From the vast possibilities of data representation formats in Vowpal Wabbit, we have chosen to use the simplest representation. We did not use namespace tags, per namespace weights, or per instance weights. We have also decided not to weight particular features. The data format is then simply:

$1 \mid f_1 \ f_2 \ f_3 \ NE \ f_4 \ f_5 \ f_6$

Dataset modification was presented in a previous section under Feature Engineering. After the best performing feature set was chosen, we used the dataset to tune parameters and the learning method to select the best performing combination. The machine learning method is divided into two stages – training and testing. To train the classifier we have chosen different parameter schemes from the options implemented in Vowpal Wabbit. We have trained several models using different loss functions, regularization and learning rates. In the experiments depicted in the following chapter, we tried four different loss functions: Squared, Quantile, Logistic, and Hinge. Additionally, we have tuned the Lasso and Ridge regularization parameters by adopting the --*l1* and --*l2* parameters for each of the loss functions. We have tried different learning rates within the range of the default value as well as different values for the *power_t* parameter. All of the feature engineering and parameter tuning experiments were tested on the development dataset.

4 Results

The results are presented separately for the Distant Supervision method and the Snowball system. The first section is devoted to the experiment results on the development dataset, including Feature Engineering and Parameter Tuning experiments. The second section presents the results on the test dataset, using the best performing models from the first section.

4.1 Results on Development Dataset

4.1.1 Distant Supervision

The first set of experiments is designed to select the best feature set for the learning algorithms. We have used the default settings in Vowpal Wabbit. The default loss function is Squared, without Lasso and Ridge regularizations. The default learning rate is 10 and the default *power_t* is 0.5. The models were built with one pass through the data. As noted before, Vowpal Wabbit is an online learning system. In online learning, the features weight vector is updated for each example. This makes the algorithm very sensitive to the order of the data instances. To reduce the effect of data ordering, all the experiment were run 5 times on differently randomized datasets. The results reported are averaged from the 5 runs.

Experiments 1-10 contain datasets with different window size for the context surrounding the named entity. A window size of 1 takes one token on each side of the named entity. The feature vector also contains the named entity itself. We have also added an experiment with only the left context words included as features. Another experiment does not include the named entity in the feature vector.

Table 13. Window size experiments for Distant Supervision

Experiment	Window Size	Recall	Precision	F-score
1	1	86.41	92.39	89.29
2	2	86.41	92.91	89.50
3	3	**89.66**	**90.55**	**90.08**
4	4	88.57	91.23	89.87
5	5	85.03	89.28	87.07
6	6	86.49	89.33	87.86
7	7	84.53	86.13	85.27
8	8	83.84	87.24	85.45
9	9	83.15	84.43	83.72
10	10	81.51	88.27	84.68
11	3 only left side	87.96	92.24	90.03
12	3 without NE	67.64	79.23	72.93

The best results in this set of experiments are with the smaller window size. Three context words on each side are enough for the algorithm to learn if the sentence describes the *country-of-origin* relation. It makes sense since the word right next to the named entity is semantically the most informative. The named entity itself proved to be an important feature for the classification. The F-score of the experiment without NE is almost 20 lower than the same experiment with the NE. Taking only the left context words, yielded very similar results, with only 0.05 points difference in F-score. Nevertheless, we decided to continue with the experiments using the highest achieving model, with 3 words from both sides of the context.

In the next experiment set, we tested different stop-word lists. Complete refers to the full stop-word list; Reduced refers to the stop-word list without relevant prepositions and adverbs; Domain refers to the domain specific stop-word list.

Table 14. Stop-words removal experiments for Distant Supervision

Experiment	Stop-word list	Recall	Precision	F-score
13	No stop-word list	**89.66**	**90.55**	**90.08**
14	Complete	85.82	92.32	88.93
15	Reduced	86.92	91.68	89.23
16	Domain	83.94	91.16	87.34
17	Complete + Domain	84.14	93.13	88.39
18	Reduced + Domain	84.46	92.55	88.31

The removal of words from the domain specific stop-words list did not improve the performance of the algorithm. All three experiments that include domain list removal did not over-perform the model without stop-word list. All the experiments in this set produced higher precision than the model without stop-words removal. On the other hand, the recall decreased in those models. We do not use stop-word lists in the subsequent experiments, given that the F-score did not improve after stop-word removal.

In the following set of experiments we have normalized the data by exchanged numeric values and punctuation by PUN and NUM respectively. Additional setting includes the removal of numbers and punctuation.

Table 15. Numbers and punctuation normalization experiments for Distant Supervision

Experiment	Experiment setting	Recall	Precision	F-score
19	No normalization	**89.66**	**90.55**	**90.08**
20	Punctuation replaced	86.47	90.84	88.58
21	Punctuation removed	87.31	91.65	89.40
22	Numbers replaced	87.57	92.08	89.75
23	Numbers removed	87.31	91.28	89.24
24	Both replaced	87.96	91.41	89.61
25	Both removed	87.51	92.13	89.74

Punctuation and numerical values usually do not contribute the text classification task since their semantic input is poor. On the other hand, removal of those features did not improve the performance.

The next set of experiments tests whether linguistic features contribute to performance. By linguistic features, we mean linguistically driven modifications of the word tokens. We performed morphological analysis to extract lemmas and part of speech tags. We have also stemmed the tokens. It is a common claim that feature vectors represent one term frequency at each position (Salton, et al., 1975). In this work we tried a different approach where each term is represented at several points in the vector. Instead on merely changing the representation of one token from word form to lemma or stem, we add both stems and lemmas to the bag of words language model. In Mintz et al. work (2009), they constructed complex features, representing the whole pattern as one feature. We are taking the opposite approach, breaking the representation of the pattern to many separate features. We have also combined lemmas with POS tags and Stems with POS tags as one feature.

Table 16. Linguistic features experiments for Distant Supervision

Experiment	Experiment setting	Recall	Precision	F-score
26	Word forms	89.66	90.55	90.08
27	Lemmas	88.60	92.85	90.64
28	POS tags	83.17	91.32	87.01
29	Stems	87.31	91.84	89.51
30	Lemmas - POS tags	**91.00**	**93.92**	**92.42**
31	Stems - POS tags	87.96	92.91	90.34
32	Lemmas +Stems	87.89	90.18	89.01

As expected, lemmas contribute the performance of the system. The model can generalize over several word forms using their lemmatized representation. The combination of lemmas with POS tags achieves the highest F-score of 92.42. It is surprising that stemming did not have a similar effect on the performance.

We have added another set of experiments using n-gram and skip-gram models. An n-gram model takes *n* consecutive words and concatenates then into one feature. Skip-gram model creates n-gram features by combining *n* words separated by *k* words from each other. The best model so far was created as a unigram model (1-gram) of lemma-tag terms. In the following experiments, we used the lemma-tag features and created bigram and trigram models from them. Similarly, we created skip-bigram and skip-trigram models with skipping over one word and two words.

Table 17. N-gram and Skip-gram experiments for Distant Supervision

Experiment	Experiment setting	Recall	Precision	F-score
33	Unigram	**91.00**	**93.92**	**92.42**
34	Bigram	85.24	91.46	88.24
35	Trigram	86.73	91.40	89.00
36	Skip 1 bigram	90.42	93.54	91.94
37	Skip 2 bigram	89.06	94.16	91.51
38	Skip 1 trigram	87.64	94.32	90.84
39	Skip 2 trigram	89.84	93.92	91.82

Bigram and trigram models did not improve the performance over the unigram model. Skip-gram models provided better results than the regular n-gram models, but not exceeding the unigram performance.

Next, we performed another set of experiments for parameter tuning. Table 18. includes results of the best performing sets of parameters per loss function. All the experiments were run on the following feature set: 3 tokens on each side on the named entity including the NE; the tokens are lemmatized, and a POS tag is attached to every lemma (lemma-tag); without normalization of punctuation and numbers; without stop-words removal. The tuned parameters include: loss function {squared, hinge, quantile, logistic}; L1 and L2 regularization {0, 0.01, 0.001, 0.0001, 0.00001}; Learning rate {1 5 10 15 50}; Power_t {0 0.1 0.3 0.5 0.7 0.9 1}. We have experimented with a different number of passes through the data. We concluded that the number of passes in the range of 8-12 achieves relatively close results. Therefore, we tune the other parameters with a fixed value of 10 for the --passes parameter. We have tuned the parameter in a matrix-like fashion. We defined several categorical values for each parameter orienting around the default value. We then ran the algorithm with all possible combinations of those values. Again the data was randomized 5 times and the results are averaged over the different datasets.

Table 18. Parameter tuning for Distant Supervision

Exp	Loss Func.	L1	L2	Power t	Learn Rate	Recall	Precision	F-score
40	Logistic	0.001	0	0.3	10	**100**	**98.43**	**99.21**
42	Squared	0.001	0.0001	0.9	50	100	97.22	98.59
44	Quantile	0.01	0.001	0.3	5	100	97.06	98.50
46	Hinge	0.01	0	0.1	5	100	97.06	98.50

Best performing model for the *country-of-origin* relation extraction using the logistic loss function has been achieved by using L1 regularization with lambda value 0.001, 0.3 power on the learning rate decay, and default lambda learning rate parameter. With those parameters, the logistic model yields 99.21 F-score.

All the loss functions produced well performing models. There is no significant difference between the results in Table 18. Actually, we did not observe consistent different between the results produced by specific values in any of the parameters. All values in different combinations produced results which are close to the best performing model.

4.1.2 Snowball System

The method used in the Snowball system is very different from the Distant Supervision method. We decided to tune the similarity threshold and the overlap threshold first since with some parameters the system fails to perform. For the first parameter tuning, we used 3 tokens from each side of the named entity as features. After finding the best values for those parameters, we test the system on other feature sets (similarly to the Distant Supervision feature engineering). We then performed another set of experiments to tune the system again with the selected feature set, adjusting the term weighting scheme.

In Table 19. we present the experiments of the initial parameter tuning. In Distant Supervision experiments, we chose the best model by F-score, as Recall and Precision are more or less balanced in all of the experiments. In the Snowball system, we concentrated on Precision scores when selecting the best performing model. The recall scores in the above experiments are significantly higher than the precision scores. In fact, as experiment 16 shows, if we the recall is almost 100, we get a baseline precision of 29.22. We could achieve the same precision and recall scores if we would simply label all instances with 1. In Snowball, we want to achieve a higher percentage of correctly labeled data instances, even if the recall scores are decreasing. The best performing model after the first parameter tuning is with similarity threshold 0.2 and

overlap threshold of 5, achieving 30.83 precision score. Note that this is only a minor improvement over the baseline precision.

Table 19. Similarity and overlap parameters tuning for the Snowball system

Experiment	Similarity	Overlap	Recall	Precision	F-score
1	0.11	10	91.33	30.09	45.27
2	0.11	15	88.33	30.22	45.03
3	0.12	5	89.38	30.11	45.05
4	0.12	15	84.44	30.27	44.57
5	0.13	5	88.41	30.12	44.93
6	0.13	15	83.79	30.30	44.51
7	0.13	20	83.14	30.66	44.80
8	0.14	5	87.11	30.13	44.77
9	0.14	15	83.55	30.36	44.53
10	0.14	20	81.93	30.64	44.59
11	0.15	5	86.22	30.25	44.79
12	0.15	15	82.66	30.28	44.33
13	0.15	20	80.87	30.57	44.36
14	0.16	5	84.93	30.28	44.64
15	0.16	20	80.23	30.54	44.24
16	**0.2**	**5**	**73.01**	**30.83**	**43.36**
17	0.01	1	95.46	29.22	44.73

With the selected overlap and similarity threshold parameters, we continued to feature engineering experiments. As in the previous Section, we first present the results of different window size feature sets.

Table 20. Window size experiments for Snowball

Experiment	Window Size	Recall	Precision	F-score
18	1	92.62	29.59	44.85
19	2	64.91	31.53	42.44
20	3	73.01	30.83	43.36
21	4	75.12	29.28	42.14
22	5	76.90	29.86	43.01
23	6	75.93	29.78	42.78
24	7	76.58	29.48	42.57
25	8	77.55	29.53	42.78
26	9	83.38	29.44	43.51
27	10	78.44	29.20	42.55
28	3 only left side	76.34	31.62	44.72
30	2 only left side	**65.56**	**31.86**	**42.88**

Similarly to the Distant Supervision results, Snowball performs better with a smaller window size. Moreover, taking only the two tokens on the left side of the NE yielded the best precision score so far: 31.86.

Next, we experimented with different stop-word lists. The following experiments were run with two tokens making the left context, ignoring the right context. We have

used the same stop-word lists as in the Distant Supervision experiments: Complete list, Reduced list and Domain list.

Table 21. Stop-words removal experiments for snowball

Experiment	Stop-word list	Recall	Precision	F-score
31	No stop-word list	65.56	31.86	42.88
32	Complete	60.53	30.21	40.30
33	Reduced	63.94	30.93	41.69
34	Domain	31.60	55.79	40.35
35	Complete + Domain	**50.81**	**58.32**	**54.31**
36	Reduced + Domain	26.42	54.79	35.65

The removal of words from the domain specific stop-words list improves the precision of the algorithm dramatically. In combination with the complete stop-word list, the algorithm achieves balanced recall and precision, with 54.31 F-score. This is the best performing setting so far. The complete stop-word list and the reduced list alone did not have such an effect on the performance.

We continue with experimentation, using the two tokens from the left, and removing stop-words from the domain and complete lists. The following set of experiments, we test the effect of punctuation and numbers normalization.

Table 22. Numbers and punctuation normalization experiments for Snowball

Experiment	Experiment setting	Recall	Precision	F-score
37	No normalization	**50.81**	**58.32**	**54.31**
38	Punctuation replaced	50.89	57.61	54.04
39	Punctuation removed	50.97	57.49	54.04
40	Numbers replaced	50.97	57.81	54.18
41	Numbers removed	30.39	55.72	39.33
42	Both replaced	30.71	55.65	39.58
43	Both removed	51.05	57.48	54.08

Punctuation and numerical values did not effect the results. In all of the experiments above, the precision stayed stable, while the recall dropped in experiments 41 and 42. Based on those results, we decided not to normalize punctuating and numerical values in subsequent experiments.

In the following set of experiments, we apply morphological analysis on the token constituting the left context. We replaced the word forms with lemmas and Stems. We have also tried to combine POS tags with lemmas and stems. The last experiment takes both stems and lemmas as features.

Table 23. Linguistic features experiments for Snowball

Experiment	Experiment setting	Recall	Precision	F-score
44	Word forms	50.81	58.32	54.31
45	Lemmas	15.33	58.94	24.33
46	Stems	19.29	59.57	29.15
47	Lemmas - POS tags	13.44	56.93	21.74
48	Stems - POS tags	19.55	58.81	29.35
49	Lemmas +Stems	**60.03**	**58.62**	**59.32**

All of the experiments performed well from a precision point of view. However, lemmas and stem on their own produced lower recall levels. The combination of stems and lemmas achieve the highest results in both recall and precision, with 59.32 F-score.

We have not implemented n-gram supporting algorithm for Snowball. In order to incorporate successfully n-gram features, we would need to adjust the seed representation and the seed enrichment method. We leave this extension for future work.

Feature engineering helped us to improve the performance of the system. We continued our experiments with term weighting parameters. The term weighting scheme is of the form $L_{ij}G_iN_j$. We have applied the same normalization to all of the vectors. We experimented with local and global weighting schemes. We first tested term frequencies, relative term frequencies, Log normalization and SQRT normalization as local weighting. Subsequently, we applied global weighting – IDF in combination with the different local weights. IDF weighting was applied on the data instances only while local weighting was applied on the seed vectors as well.

Table 24. Weighting scheme parameter tuning for Snowball

Experiment	**Weight scheme**	**Recall**	**Precision**	**F-score**
50	Tf	60.03	58.62	59.32
51	Rel	67.61	58.06	62.47
52	Log	78.98	58.00	66.88
53	Sqrt	**79.15**	**58.09**	**67.01**
54	Tf-idf	45.22	57.25	50.53
55	Rel-idf	47.20	57.93	52.02
56	Log-idf	43.67	56.46	49.25
57	Sqrt-idf	52.19	58.61	55.22

The best performing weighting scheme is SQRT local normalization without global normalization and with cosine length normalization. The best performing feature set included stems and lemmas of the two tokens from the left side of the NE. We removed stop-words from the complete general stop-word list and from the domain specific stop-word list. With those settings, we have achieved 67.01 F-score.

4.2 Results on Test Dataset

4.2.1 Distant Supervision

On the test dataset, we have tested the best performing models from the different stages of Feature engineering and parameter tuning. The results are presented in the following tables. We first tested the model with word forms, window size 3 and default Vowpal Wabbit parameters. The next well-performing model was created by changing the word forms to lemma-POS format. Subsequently, we present the results of the system using the best performing parameter sets.

Table 25. Results on the test dataset feature engineering stage for Distant Supervision

Experiment	Experiment settings	Recall	Precision	F-score
1	3 word form	86.60	92.48	89.44
2	3 Lemma-POS	88.54	92.95	90.68

Table 26. Results on the test dataset parameter tuning stage for Distant Supervision

Exp	**Loss Func.**	**L1**	**L2**	**Power t**	**Learn Rate**	**Recall**	**Precision**	**F-score**
3	Logistic	0.001	0	0.3	10	95.23	97.23	96.21
4	Squared	0.001	0.0001	0.9	50	96.87	95.55	96.19
5	Quantile	0.01	0.001	0.3	5	**1**	**97.06**	**98.50**
6	Hinge	0.01	0	0.1	5	94.88	92.05	93.44

The results on the test dataset are close to the results on the development dataset. However, the F-scores are lower using the test dataset. The best performing model is using lemma-POS features, with Quantile loss function, L1 regularization with lambda value 0.01 and L2 regularization with lambda value 0.001, the *power_t* parameter set to 0.3 and 5 as the value of lambda parameter of the learning rate. This model achieves 98.5 F-score on the test dataset.

The results show that the machine learning algorithm used for the Distant Supervision is not sensitive to the specific dataset used to test the models. The feature set that we chose and the parameters we tuned perform very well on a dataset different from the development dataset. The Distant Supervision models benefit from certain linguistic modifications such as lemmatization and part of speech tagging. We did not find other linguistic modifications such as punctuation normalization and numerical values normalization beneficial for our models. Stop-words removal did not improve the performance of the models as well. That said, we cannot conclude that those feature

modifications would not be helpful on different data or using a different method. We have also found that the method requires a rather small context window for learning. In fact, wider context window decreases the results. We conclude that the most informative context words are the ones closer to the named entity.

4.2.2 Snowball System

In this Section, we provide the results of the Snowball system on the test dataset. As in the previous Section, we tested Snowball performance in different stages of feature selection and parameter tuning.

Table 27. Results on the test dataset of the Snowball system

Exp.	Experiment setting	Recall	Precision	F-score
1	2 Left word forms; similarity 0.2; overlap 5	66.44	33.27	44.34
2	2 Left word forms; similarity 0.2; overlap 5; stop-words	29.26	57.05	38.68
3	2 lemmas & stems; similarity 0.2; overlap 5; stop-words	45.09	59.41	51.27
4	2 lemmas & stems; similarity 0.2; overlap 5; stop-words; sqrt	**51.46**	**59.09**	**55.01**

The results on the test dataset are significantly lower than the results on the training dataset. The best performing feature set and parameters achieve 55.01 F-score, 12 points less than the F-score achieved during training. The reason for the lower performance on the test dataset lays in the architecture of the algorithm. The algorithm requires that we find similar data instances to the seed patterns and extract useful features to expand the seed patterns. This makes the algorithm very sensitive to the data it runs on. Not like machine learning algorithms, we do not provide the system with many positive examples to train a model. In machine learning, after the model is trained, the classifier can evaluate new data instances individually. In the Snowball system, we do not train a model. We can tune the parameter of the system to run on new datasets, but the performance would still depend on the data the system is using for this particular run. The performance of the algorithm also depends on the size of the dataset. Snowball tries to generalize the pattern to extract as many positive examples as possible. If the dataset is too small, the generalization will be poor. It is not given that the iterative process will identify all the possible pattern that express the relation. We have tested the Snowball system on a dataset half the size of the training dataset. Therefore, it is expected that the performance will be lower.

The Snowball system showed significantly higher results using linguistically modified features, both in training and testing stages. We have used both stems and

lemmas to represent each term in the feature vector. As noted before, in this approach the feature vectors are twice as long, with more than one vector point per term. Using this representation the system achieved significantly higher results, both in training and testing. Snowball performed better after the removal of stop-words from both the standard list of stop-words and the domain specific stop-words list during training. On the test dataset, this experiment yielded better precision but lower recall. However, in subsequent experiments on the test dataset, the system achieved balanced recall and precision scores, with the removal of stop-words. The best results both in training and testing were achieved after SQRT normalization of the feature vectors.

Even though the results on the test dataset are lower than the results on the training dataset, we consider the system successful. The overall performance on the test dataset is 10 points higher than the baseline 44.85 F-score. The precision of the system is 30 points higher than the baseline. Given the results of Snowball on the test dataset, we were able to extract 51% of the positive examples with 59% correctly extracted examples.

Conclusion

In this thesis, we presented results of two methods for relation extraction applied to the business domain. The Snowball system (Agichtein & Gravano, 2000) and Distant Supervision (Craven & Kumlien, 1999; Mintz, et al., 2009) were both adapted for the unique data provided by a corporate partner. The methods were implemented to use both structured and unstructured data from the database of the company. Throughout the work, we have used the *country-of-origin* relation to demonstrating the implementation and results of the methods.

On the task of *country-of-origin* relation extraction the Distant Supervision method achieved significantly higher results than the Snowball system. On the test dataset of 1089 examples, Distant Supervision achieved 98.5 F-score. While the Snowball system achieved only 55.01 F-score on the same dataset. Given those results, we conclude that the machine learning approach is more suitable for this task. The Distant Supervision algorithm did not require a large dataset to train a well-performing model. Moreover, there are no task-specific components in this method, therefore, we expect it to perform well on other datasets and relations.

The Snowball system has an advantage over the Distant Supervision method. Except seed construction, it does not require labeled data. There is no learning involved, therefore, the algorithm can be simply applied to new datasets as is. After the seed patterns are constructed, the system will find similar patterns and extract relation pairs. In practice, though, running the system on a rather small dataset, we realized that Snowball is sensitive to the similarity and overlap threshold parameters. For both parameters, a small difference between the passed values meant a great different in the results. Tuning those parameters for a new dataset or relation would require labeled dataset for evaluation. Moreover, on such a small dataset the results of the system are not definitive, extracting only half of the positive examples, and marking half of the extracted examples incorrectly. That said, the system achieves significantly higher results over the baseline. We believe that the system would perform better on a larger dataset. As Snowball depends on having the repetitive appearances of positive examples in the data, the more data we have, the better generalization the Snowball system will achieve.

This thesis is an experimental work. We do not have previous work to compare our results to. Both methods were successfully implemented on the business domain data.

We are confident that the resulting implementation of this thesis can be applied to extract other relations, given the demanded structure, where the named entity of the extracted relation is present in both structured and unstructured data. We expect that other companies could use the algorithms as well. As part of future work, we would like to confirm this assumption by testing both methods on additional datasets of other companies, and by extracting other relations.

The performance of the algorithms depends on the quality of the data used. If the labels in the structured data are incorrect, the system will consider a positive example negative or vice versa. The quality of the text also affects the performance. Spelling mistakes might harm generalization, as might spacing errors and non-standard abbreviations. As part of future work, we want to test this effect, by standardizing the data and comparing the results to the ones achieved in this work. Another factor affecting the quality of the data is the performance of the linguistic tools used in this thesis. We have performed a small evaluation of the NameTag and Morphodita tools (Straková, et al., 2013; Straková, et al., 2014). Although the tools perform fairly well on this domain data, it is possible to improve their performance by training them on labeled domain data language models. After the tools are trained on domain data, it would be interesting to compare the performance of differently trained tool on other data from a similar domain.

Bibliography

Agichtein, E. & Gravano, L., 2000. *Snowball: Extracting relations from large plain-text collections.* s.l., ACM, pp. 85-94.

Baars, H. & Kemper, H. G., 2008. Management support with structured and unstructured data—an integrated business intelligence framework. *Information Systems Management,* pp. 132-148.

Bach, N. & Badaskar, S., 2007. A review of relation extraction.. In: *Literature review for Language and Statistics II.* s.l.:s.n.

Banko, M. et al., 2007. *Open information extraction from the web.* s.l., s.n., pp. 2670-2676.

Bejček, E. et al., 2012. *Prague Dependency Treebank 2.5–a revisited version of PDT 2.0.* s.l., s.n., pp. 231-246.

Brin, S., 1999. Extracting patterns and relations from the world wide web. In: *The World Wide Web and Databases.* s.l.:Springer Berlin Heidelberg, pp. 172-183.

Chisholm, E. & Kolda, T. G., 1999. *New term weighting formulas for the vector space method in information retrieval,* s.l.: Oak Ridge National Laboratory, US.

Cortes, C. & Vapnik, V., 1995. Support-vector networks. In: *Machine learning.* Boston: Kluwer Academic Publishers, pp. 273-297.

Craven, M. & Kumlien, J., 1999. Constructing biological knowledge bases by extracting information from text sources. *ISMB,* August, Volume 1999, pp. 77-86.

Dörre, J., Gerstl, P. & Seiffert, R., 1999. *Text mining: finding nuggets in mountains of textual data.* s.l., ACM, pp. 398-401.

Duchi, J., Hazan, E. & Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research,* Issue 12, pp. 2121-2159.

Fan, W., Wallace, L., Rich, S. & Zhang, Z., 2006. Tapping the power of text mining. *Communications of the ACM,* 49(9), pp. 76-82.

Gangadharan, R. G. & Swami, S. N., 2004. *Business intelligence systems: design and implementation strategies.* s.l., IEEE, pp. 139-144.

Grimes, S., 2014. *Text Analytics 2014: User Perspectives on Solutions and Providers,* s.l.: Alta Plana Corporation .

Joachims, T., 1998. *Text categorization with support vector machines: Learning with many relevant features.* s.l., Springer Berlin Heidelberg, pp. 137-142.

Jurafsky, D. & Martin, J. H., 2009. *Speech and Langauge Processing.* Second ed. s.l.:Pearson Education Interational.

Karampatziakis, N. & Langford, J., 2010. *Online importance weight aware updates,* s.l.: arXiv preprint arXiv:1011.

Langford, J., Li, L. & Strehl, A., 2007. *Vowpal wabbit open source project,* s.l.: Yahoo!.

Manning, C. D. & Schütze, H., 1999. *Foundations of statistical natural language processing.* s.l.:MIT press.

Mintz, M., Bills, S., Snow, R. & Jurafsky, D., 2009. *Distant supervision for relation extraction without labeled data.* s.l., Association for Computational Linguistics, p. 1003–1011.

Porter, M. F., 2001. *Snowball: A language for stemming algorithms,* s.l.: s.n.

Salton, G., Wong, A. & Yang, C.-S., 1975. A vector space model for automatic indexing. *Communications of the ACM,* pp. 613-620.

Ševčíková, M., Žabokrtský, Z. & Krůza, O., 2007. Named entities in Czech: annotating data and developing NE tagger. *Text, Speech and Dialogue. Springer Berlin Heidelberg*, pp. 188-195.

Straková, J., Straka, M. & Hajič, J., 2013. *A New State-of-The-Art Czech Named Entity Recognizer.* místo neznámé, Springer Verlag, Berlin / Heidelberg, pp. 68-75.

Straková, J., Straka, M. & Hajič, J., 2014. *Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition.* Baltimore, Maryland, Association for Computational Linguistics, pp. 13-18.

Vlachos, A. & Clark, S., 2014. *Application-Driven Relation Extraction with Limited Distant Supervision.* s.l., s.n., pp. 1-6.

List of Tables

List of Figures

List of Abbreviations

RE	Relation Extraction
NE	Named Entity
NER	Named Entity Recognition
BI	Business intelligence
NLP	Natural Language Processing
GD	Gradient Descent
SVM	Support Vector Machine
POS	Part of Speech
VW	Vowpal Wabbit

Attachment

Attachment 1 – Full hierarchy extracted by NameTag (Ševčíková, et al., 2007).