TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER'S THESIS no. 1321

A Joint Model for Named Entity Relation Extraction

Dino Radaković

Zagreb, July 2016

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 11. ožujka 2016.

Predmet: **Analiza i pretraživanje teksta**

DIPLOMSKI ZADATAK br. 1321

Pristupnik: **Dino Radaković (0036463361)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Združeni model za ekstrakciju relacija između imenovanih entiteta**

Opis zadatka:

Imenovani entiteti (osobe, organizacije, lokacije i sl.) u tekstu se pojavljuju u različitim međusobnim relacijama (npr. poznanstvo, smještenost i sl.). Ekstrakcija relacija između imenovanih entiteta važan je problem semantičke analize teksta. Zadatak se tipično rješava metodama strojnog učenja, i to klasifikacijom na razini para entiteta, a na temelju značajki teksta koje upućuju na semantičku relaciju. Međutim, takva lokalna obrada ne uzima u obzir odnose i ograničenja koja vrijede između više parova entiteta. Modeli koji takva ograničenja uzimaju u obzir temelje se na združenom zaključivanju i naknadnom ispravljanju oznaka lokalnih modela, ili na združenom učenju.
U okviru diplomskoga rada potrebno je proučiti postojeće pristupe ekstrakciji relacija te istražiti značajke korištene za taj zadatak. Proučiti modele združenog zaključivanja, s naglaskom na postupak cjelobrojnog linearnog programiranja, te modele združenog učenja, s naglaskom na SEARN (Daumé III i dr., 2006.). Razraditi model za ekstrakciju relacija u tekstovima na hrvatskome jeziku i razviti odgovarajuću programsku implementaciju. Provesti označavanje prikladnog skupa podataka za učenje i vrednovanje modela. Provesti iscrpno eksperimentalno vrednovanje modela na ispitnim skupovima podataka te analizu pogrešaka. Radu priložiti izvorni i izvršni kod razvijenog sustava, skupove podataka i programsku dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 1. srpnja 2016.

Mentor:

Doc. dr. sc. Jan Šnajder

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Prof. dr. sc. Siniša Srbljić

CONTENTS

1. Introduction

Information extraction, a subfield of natural language processing, has been receiving a lot of attention in the current decade. This attention mainly stems from recent advances in applied machine learning, such as efficient implementations of support vector machines (Fan et al., 2008), deep neural networks (Collobert and Weston, 2008), and a steady Moorean increase in widely available computational power.

Information extraction is usually defined as recognition and classification of predefined types of entities, relations or events in text, written in some natural language (Manning and Schütze, 1999). The task itself can be executed manually. However, manual information extraction requires an expensive investment in terms of time and manpower, and as such does not scale well with respect to the projected growth of publicly available textual data (Gantz et al., 2010). Machine learning methods are therefore employed for the purpose of automation, which entails scalability. Such problems' instances may be complex structures, rather than relatively simple instances of straightforward classification problems. An example of a complex structure would be a natural language parse tree, as opposed to the example of sentence splitting, which can directly be reduced to the classification problem of determining sentence boundaries, for many natural languages, such as Croatian, English, and Italian, among others. Problems whose instances are structured are often decomposed into substructures or segments of the given structure, which can then be solved using relatively simpler machine learning algorithms. In terms of named entity relation extraction, this amounts to classifying each edge (or lack thereof) of the entity-relation graph individually, as opposed to trying to learn and predict graphs altogether. This, however, often does not take into account the conditional statistics inbetween the segments themselves. For example, edges from vertex a to vertex b and from vertex b to vertex c in a certain graph might imply a higher probability of the existence of an edge between vertex a and vertex c, as would be the case when attempting to enforce transitivity in the context of coreference resolution.

Recent advances allow researchers to tackle more of such problems in their original

form using structured prediction, rather than splitting them into separate classification problems, avoiding the irreversible loss of information, written in the form of interdependencies between the components themselves. Structured prediction techniques based on joint learning help researchers tackle structured prediction without having to nullify the interdependence of structures' individual components. Classes of machine learning algorithms, such as search-based learning (Daumé III et al., 2009), (Krishnamurthy et al., 2015) and graphical models, have been surging due to training of corresponding models becoming computationally feasible in terms of practical applications. They represent a more consistent approach to learning the whole joint distribution over structured problems, as opposed to arbitrary traditional decomposition-unification schemes relying solely on individual classifiers. Substantial joint inference methods in the form of probabilistic models used to unify individual classifiers' outputs with the purpose of approximating the joint distribution prior to its implicit decomposition have yet to surface, as the currently available models do not seem to provide significant improvements in terms of F-scores (Roth and Yih, 2004), Culotta and Sorensen (2004).

In this thesis, we apply a joint learning model called SEARN to named entity relation extraction, an increasingly significant real-world problem. The purpose of our work is to probe and analyze the potential applications of joint learning applied to relation extraction, specifically for Croatian language. We start by describing named entity relation extraction and related research in the field. We present a novel dataset for relation extraction, consisting of manually annotated Croatian newswire text. We then proceed by designing and implementing both syntactic and semantic feature-based models: a baseline rule-based model, a simple (non-structured) SVM-based model and, finally, a model based on SEARN, evaluated on the introduced dataset. Among the available joint learning models, we chose SEARN (Daumé III et al., 2009), a search-based model which draws inspiration from research in reinforcement learning. An efficient implementation of SEARN and related search-based models (Langford et al., 2011) served as the primary framework upon which a joint learning model was built, as proof of concept.

The implemented models are evaluated on the introduced dataset using standard F-scoring and compared with theoretical upper bounds derived from inter-annotator agreement metrics.

2. Named Entity Relation Extraction

2.1. Named Entity Recognition and Classification

2.1.1. Definition

Named entity, as a concept, lacks a standard definition. The possibly first formal mention of the term occurrs in Grishman and Sundheim (1996) and with it the relatively new task of named entity recognition and classification (NERC). As parts of some text, named entities are (possibly non-consecutive) k-grams that typically refer to real-world entities, such as people, locations, organizations, etc.

By definition, attempting to solve NERC usually requires handling the issue of tokenization. This is often solved using finite state machines or rule-based models with manually designed rules, sometimes using statistical models, as described by Bird (2006), and as such will not be discussed as part of this thesis.

NERC itself can be expressed in form of two consecutive steps:

1. Given some text, identify the tokens corresponding to named entities, where a single named entity may be composed of multiple tokens

2. Having identified the named entities, decide on one of the predefined classes for each one of them.

For example, suppose we have a NERC model that detects entities and labels each one of them as either "Person" or "Location", both labels bearing purely etymological semantics.

Consider the following sentence:

"Marcus Aurelius and Lucius ruled Rome as co-emperors for 8 years."

The sentence consists of the following tokens:

Marcus[1]	Aurelius[1]	and	Lucius[2]	ruled	Rome[3]	as	co-emperors	for	8	years	.

In the given tokenized representation of the sentence, entities consist of gray tokens – tokens with matching superscript indices being associated with the same corresponding named entity. There are three named entities in the sentence: **Marcus Aurelius** (1), **Lucius** (2), and **Rome** (3), whose types our model would ideally label as *Person*, *Person* and *Location*, respectively.

NERC is often considered to be an unavoidable precusor to the task of named entity relation extraction, due to the latter usually requiring named entities during both training and testing phases, if posed as a machine learning problem.

2.1.2. Solving NERC

Named entity recognition and classification is almost exclusively approached as a supervised learning problem. Given positive and negative examples of named entities within text, the goal is to learn a model that can then hopefully recognize and classify named entities in general text, or at least in text within the same domain or one written in similar fashion, depending on the purpose of the application.

Early research was geared towards discriminative models, such as decision trees and SVMs (Nadeau and Sekine (2007)), but soon shifted towards generative graphical models – namely, hidden Markov models and linear-chain conditional random fields (CRFs) (Tjong Kim Sang and De Meulder (2003)). For a given classification problem consisting of assigning classes $y \in \mathcal{Y}$ to instances $\mathbf{x} \in \mathcal{X}$ (data mapped to feature space), generative approaches involve learning distributions $p(x|y)$ and $p(y)$ and modeling $p(y|\boldsymbol{x})$ using Bayes' rule, as $p(y|\boldsymbol{x}) \propto p(\boldsymbol{x}|y) \cdot p(y)$, assuming an uniform prior $p(\boldsymbol{x})$. On the other hand, discriminative methods are direct attempts at modeling $p(y|\boldsymbol{x})$. Generative models enable generating data by sampling $p(x|y)$ (hence the name), while discriminative ones establish boundaries between classes in feature space, based on maximum likelihood estimation: $\hat{y} = \arg\max_y p(y|\boldsymbol{x})$.

Probabilistic graphical models in particular form the basis for current state of the art NERC, for both English (McCallum and Li (2003)) and Croatian (Glavaš et al. (2012)). Named entity recognition and classification is often treated as a sequence labeling problem, where the task reduces to correctly labeling a subsequence of tokens referring to named entities within given text.

Linear chain CRFs for named entity recognition are typically trained over *sliding windows*, contiguous subsequences of tokens mapped to feature space. Assuming a

NERC model that can, among others, detect references to persons in text and label them as *Person*-type named entities, consider the following sentence:

"*The whole world watched* ***Neil Armstrong*** *step on the surface of the moon.*"

The occurrence of "Neil Armstrong", a *Person*-type entity may be inferred from the surrounding non-stopwords, their respective capitalization, part-of-speech tags and possibly positions within the sentence's dependency tree, among others. Isolating the preceding and succeeding words relative to the named entity, "*watched*" and "*step*", leads to a conclusion that some sort of entity was being *watched*, and that it had *stepped*. Since *stepping* would typically be associated with people, rather than other types of named entities, such as organizations, a keen observer could deduce that the entity refers to a person, which ultimately leads to the conclusion of a *Person* named entity being written between the two analyzed lexemes. On the other hand, the named entity, being a noun and the object of sentence, suggests a preceding verb, among other constituents of the sentence.

However, as opposed to discriminative models, conditional random fields take into account the statistical implications of the proximity of the named entity to those words, which is especially indicative for part-of-speech tags. For example, a named entity being a subject in a sentence significantly influences the form of neighboring parts of speech – the subject could very well be the entity behind a deed, written as a verb, which influences some object in the sentence, thus restricting the form of both verb (i.e., its person, gender and number in the context of Croatian language).

Undirected graphical models, CRFs being one of them, generally do account for this sort of mutual conditioning, as they are localized approximate representations of joint probability distributions of both observed labeling problem instances ("*central*" token labels) and labels of previously labeled tokens in their close proximity, typically ones preceding the instance (Murphy, 2012).

Even though the use of conditional random fields for named entity recognition and classification is mostly restricted to linear chain CRFs, it does represent a step towards more complex joint learning techniques and structured prediction. Unlike models that rely on performing classification on individual tokens and their respective features, joint aproaches treat the subsequences of tokens belonging to named entities spanning throughout given text as structures of interdependent labels, rather than as sets of disjoint classification problems.

2.2. Named Entity Relation Extraction

2.2.1. Definition

Named entity relation extraction is a natural extension to named entity recognition and classification. The task may be defined as follows: Given text $\mathcal{T}$ with labeled occurrences of named entities, identify and classify relations of subsets of entities, based on a chosen entity-relation scheme.

The task was first mentioned in Grishman and Sundheim (1996), being recognized as a potentially challenging, yet lucrative unsolved problem in the field of information extraction.

Assuming pairwise relations, such that relation represents a semantic relationship between two named entities, let $\mathcal{L}_\mathcal{R}$ be a set of relation types (i.e., *Located-in*, *Acquaintance-of*, etc.), $\mathcal{L}_\mathcal{E}$ a set of named entity types (i.e., *Person*, *Organization*, etc.), $\mathcal{E}$ a set of named entities, where $e_i^p \in \mathcal{E}$ would be a named entity of type $p \in \mathcal{L}_\mathcal{E}$ and $\mathcal{R}$ a set of named entity relations. $r_{i,j}^l \in \mathcal{R}$ represents a relation of type $l \in \mathcal{L}_\mathcal{R}$, between entities $e_i^p \in \mathcal{E}$ and $e_j^q \in \mathcal{E}$, $p, q \in \mathcal{L}_\mathcal{E}$. Relation types tend to impose restrictions on entity types, which can be formally written in form of a mapping $\rho : \mathcal{L}_\mathcal{E} \times \mathcal{L}_\mathcal{E} \to 2^{\mathcal{L}_\mathcal{R}}$, such that $\forall r_{i,j}^l \in \rho(p,q) | e_i^p, e_j^q \in \mathcal{E}$, where $p, q \in \mathcal{L}_\mathcal{E}$ and $l \in \mathcal{L}_\mathcal{R}$.

This thesis primarily focuses on a specific class of pairwise relations satisfying the following properties:

1. Symmetry: $r_{i,j}^l \equiv r_{j,i}^l \quad \forall i, j = 1 \ldots |\mathcal{E}| \quad \forall l \in \mathcal{L}_\mathcal{R}$
2. Non-reflexivity: $r_{i,i}^l \notin \mathcal{R} \quad \forall i = 1 \ldots |\mathcal{E}| \quad \forall l \in \mathcal{L}_\mathcal{R}$
3. Type determinism: $|\rho(p,q)| \leq 1 \;\; \forall p, q \in \mathcal{L}_\mathcal{E}$

The problem can be thought of as connecting vertices, represented by named entities, of an initially null graph with edges corresponding to relations between those entities. The output of a relation extraction model is therefore an *entity-relation graph*, whose structure has been derived from provided features.
A named entity relation is thought to be statistically conditioned on other named entity relations as well. Consider transitivity as an example of such interdependence:

"***Mark** used to work for **AcmeCorp**, where he answered directly to **Bob**.*"
"***Bob**'s work at **AcmeCorp** never earned him the respect of **Joe**, his older brother.*"

The first of the examples above showcases transitivity. *Mark* (a person) working at *AcmeCorp* (an organization), coupled with *Bob* (also a person) being his acquaintance

(as his superior) *transitively* implies the latter being an employee of the same organization as well, based on the structure of the sentence and the use of certain terms, such as "*where*".

The second example, however, shows the contextuality of such transitivity-based relation cliques – in spite of *Bob* and *Joe* being acquainted by the means of brotherhood, the former seems to be an employee of *AcmeCorp*, while is not a single indication within the sentence pointing towards the latter being an employee of the same organization as well.

While current state-of-the-art named entity relation extraction algorithms (Culotta and Sorensen (2004)) resort to classifying each possible edge of the entity-relation graph individually, by attempting to measure the likelihood of each pair of named entities corresponding to some named entity relation, not much research exists that focuses on the problem in terms of structural, rather than pairwise prediction – attempting to jointly learn and output whole graphs, rather than sequences of individually predicted edges may prove to be a more natural approach to solving named entity relation extraction.

2.2.2. Solving Relation Extraction

Support Vector Machines

Early research in named entity relation extraction focused primarily on rule-based methods, such as finite state machines described in (Byrd and Ravin (1999)). The advent of practically applicable support vector machines greatly revolutionized relation extraction (Zelenko et al. (2003)).

Support vector machines (SVMs), also known as support vector networks, are linear discriminative models first described in Cortes and Vapnik (1995) that allow for binary classification of feature vectors corresponding to classification problem instances. Support vectors classify data by splitting the feature space into two half-spaces – one ideally being the space of all positively classified problem instances, the other being the feature half-space of negative ones. For example, assuming a model used to classify the existence of named entity relations between pairs of entities, half of the feature space would correspond to entity pairs within a relation, the other half to relationless pairs of named entities.

Support vector machines are based on the *maximum margin* criterion, meaning that finding the optimal separation of k-dimensional feature half-spaces amounts to finding a $(k-1)$-dimensional hyperplane such that the distance between the closest problem instance and the hyperplane is maximized, with respect to positive and negative instances falling into their respective half-spaces.

A canonical linear SVM is described in form of the following equation:

$$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + w_0 \tag{2.1}$$

In equation 2.1, $\boldsymbol{x} \in \mathbb{R}^n$ represents a single classification problem instance, encoded as a n-dimensional vector, $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^m$ denoting the feature mapping function, which transforms the classification problem into its corresponding m-dimensional feature vector. Equation $h(\boldsymbol{x}) = 0$, that is $\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) + w_0 = 0$, uniquely determines the separation hyperplane. The scalar product of the normal of the separating hyperplane, $\boldsymbol{w}$, and the feature vector of the given classification problem, $\boldsymbol{\phi}(\boldsymbol{x})$ assigns $\boldsymbol{x}$ to one of the half-spaces, classifying x as either a positive or negative sample, depending on the sign of the scalar product. This forms the basis of prediction using support vector machines, as the decision can be formally expressed using the sign function:

$$y = sgn(h(\boldsymbol{x})) \tag{2.2}$$

Based on equation 2.2, the set of labels for an individual support vector machine is implied to be $\{+1, -1\}$.

Maximum margin optimization can be formally written as follows:

$$\arg\max_{\mathbf{w}, w_0} \left\{ \min_i \left\{ y_i(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{\phi}(\boldsymbol{x_i}) + w_0) \right\} \right\} \tag{2.3}$$

Expression 2.3 is the *primal* (directly expressed) form of the optimization problem. The Lagrangian dual form of the same problem, however, proves to more convenient in terms of time complexity required to find the optimal solution of this convex optimization problem (Sasane and Svanberg (2014)):

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x_i})^T \boldsymbol{\phi}(\boldsymbol{x_j}) \\ \textit{w. r. t.} \quad & \alpha_i \geq 0, \qquad\qquad 1 \leq i \leq N,\ i \in \mathbb{Z} \\ & \sum_{i=1}^{N} y_i \alpha_i = 0 \end{aligned} \tag{2.4}$$

Besides newly introduced Lagrangian multipliers, α_i, equation 2.4 also holds the definition of a kernel function, in the form of scalar products of feature vectors:

$$\kappa(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x'}) \tag{2.5}$$

Intuitively, the kernel function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ (2.5) can be thought of as a similarity metric for feature vectors – the smaller the kernel function value for two feature vector, the more similar the two compared classification problems, $boldsymbolx$ and $\boldsymbol{x'}$ are considered to be.

$$\boldsymbol{K} = \begin{pmatrix} \kappa(\boldsymbol{x_1}, \boldsymbol{x_1}) & \kappa(\boldsymbol{x_1}, \boldsymbol{x_2}) & \dots & \kappa(\boldsymbol{x_1}, \boldsymbol{x_N}) \\ \kappa(\boldsymbol{x_2}, \boldsymbol{x_1}) & \kappa(\boldsymbol{x_2}, \boldsymbol{x_2}) & \dots & \kappa(\boldsymbol{x_2}, \boldsymbol{x_N}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x_N}, \boldsymbol{x_1}) & \kappa(\boldsymbol{x_N}, \boldsymbol{x_2}) & \dots & \kappa(\boldsymbol{x_N}, \boldsymbol{x_N}) \end{pmatrix} \tag{2.6}$$

This is a particularly powerful tool because it allows for an implicit and functional way of expressing the notion of *similarity*, as opposed to computing the whole Gram matrix, which can be expressed in terms of pairwise kernel distances of individual problem instances.

Perhaps the simplest kernel function would be the cosine between feature vectors as seen in 2.5.

More complex kernel functions are the key to current state of the art SVM-based named entity recognition model, relying on heavy use of dependency trees – syntactic parse trees that represent dependencies between tokens in a given text. These *dependency tree kernels* produce a measure of similarity for a given pair of named entities, based on their respective dependency subtrees, augmented with lexical and morphosyntactic features, such as part-of-speech tags, frequent keywords and other linguistic patterns. The kernel distances are computed in a dynamic programming fashion – comparing pairs of subtrees of nodes recursively and only once for each pair due to memoization of already computed comparisons. The time complexity of the described algorithm is $\mathcal{O}(n^2)$, where n is the number of nodes in the larger of two compared dependency trees, assuming contiguous comparisons. Culotta and Sorensen (2004) introduce sparse kernels and compare trees by means of generalized non-contiguous subsequence comparison. While sparse kernels seem to perform better in practice, because of their ability to measure various subtree alignments, they do come at a price of $\mathcal{O}(n^4)$, again with the use of a leaf-first dynamic programming algorithm.

Alternatives

While dependency tree kernel support vector machines represent the current pinnacle of research in named entity relation extraction, there have been significant attempts at tackling named entity relation extraction using various other algorithms.

An early exemplary approach not reliant on SVMs is described by Tjong Kim Sang and De Meulder (2003), which presents a successful application of a maximum entropy classifier to relation extraction. The authors did recongize the importance of dependency tree features, leading to features similar to those introduced by Culotta and Sorensen (2004), albeit in a different context, and their model also operates on the pairwise classification level, classifying each potential named entity relation individually. Further research in maximum entropy classifiers for relation extraction was overshadowed by superior results produced by kernel methods (Zelenko et al., 2003).

A pattern-based algorithm called *Snowball* has been proposed by Agichtein and Gravano (2000), based on *seeding*. The algorithm starts with a relatively small number of manually selected pairwise named entity relations and extracts more relations from given text based on regular expressions as well as prefix and suffix oriented rules applied on the set of already extracted relations, in order to generate new patterns. The set of extracted relations initially consists of the *seed*. The patterns are then used to match and extract more named entity relations. The two steps are repeated until either no more relations can be matched or until a certain expansion threshold (number of extracted relations) has been met. The basis of this technique is known as DIPRE, introduced by Brin (1998), and has so far been successful on a limited number of named entity relations, such as authorship and location. We conjecture that rule based approaches, even advanced ones, such as DIPRE and Snowball tend to work on relations for which syntactic properties of the underlying text are sufficiently informative, as opposed to more semantic relationships, i.e., allegiance and influence. The main advantage of Snowball is its relatively low requirement for manually annotated data. The disadvantages are sensitivity to parameters (expansion control) and the content of the seeding set, which greatly influences the quality of generated patterns and matched relations.

While Snowball does focus on singled out named entity pairs during the matching phase, it does generate new patterns based on previously matched relations, which implies a notion of conditional influence on potential matches, given previously matched

relations.

2.2.3. Improving the Results

Joint Inference

As mentioned previously, methods based on classifying each entity pair individually do come at the price of disregarding influence some classified relations have on the class probability distributions of others, which are yet to be classified, and vice versa. An attempt to compensate for this loss of information surfaced in Roth and Yih (2004). The proposed algorithm assumes the ability to solve *integer linear programming* (ILP) problems – a class of linear programming problems with the additional condition of problem variables being integers. Unlike plain linear programming, ILP problems are NP-complete and as such typically require superpolynomial discrete optimization on top of convex optimization algorithms used for solving their plain counterparts (Sasane and Svanberg (2014)).

The idea can be summarized as follows:

1. Recognize and classify named entities in text
2. Extract relations in text using a pairwise learner (e.g., a dependency kernel SVM)
3. Establish a policy for inter-relation conditioning
4. Formulate an ILP problem based on (1), (2) and (3) so as to optimize for minimal loss
5. Solve (4) and re-label named entity relations and types according to (5)

A consistent ILP formulation of the problem is required to perform *joint inference* by adjusting named entities and relations after the initial predictions, which invariably ignore statistical information by assuming conditional independence of not only individual relation predictions but also the independence of named entity recognition and classification with respect to relation extraction.

Inter-relation conditioning is usually binary and used to enforce semantic restrictions required by certain named entity relations by adding a term of positive infinity the loss minimization problem, should the corresponding restriction be broken.

As much as joint inference through integer linear programming has been empirically shown to improve results in terms of $\mathbf{F}_1$scores, such improvements have so far been exclusively in the low single digit percentages for named entity relation extraction, as published by Roth and Yih (2004), which is why this approach was considered, but ultimately not included in our research.

Practical Considerations

In order to make relation extraction more convenient and usable in real-world environments, there are two more tasks that may be solved: *coreference resolution* and *entity linking*.

Named entity linking is the task of determining real-world named entities corresponding to named entity mentions in text. This involves an external knowledge base and matching algorithms. Matching namded entity mentions to entries may be done using string similarity based algorithms. However, many mentions are influenced by contextual information. For example, *Pope* could refer to *Pope Benedict XVI* or *Pope Francis*. This motivated machine learning based approaches, such as latent topic modeling described by Han and Sun (2012).

Coreference resolution means inferring that different named entity mentions, as recognized by a NERC algorithm, refer to the same real-world entity. For example:

"***President George W. Bush*** (Person) *is set to speak at the summit in November.* ***Bush*** (Person) *is expected to address the latest geopolitical questions in the wake of recent terror attacks*."

Deducing that *President George W. Bush* and *Bush* refer to the same person may seem trivial for a human reader, while posing a problem for automation. In this case, one might simply devise a substring-checking algorithm to match named entities accordingly. However, suppose that *George H. W. Bush* is mentioned in the text as well.

More issues for coreference resolution arise in highly inflective languages, such as Croatian. Consider the following translation of the previous example:

„***Predsjednik George W. Bush*** (Person) *će održati govor na summitu u studenom. Za* ***Busha*** (Person) *se očekuje da će osloviti najnovija geopolitička pitanja, povodom nedavnih terorističkih napada*."

Busha is a case-inflected form of *Bush* and as such requires more advanced string matching techniques or such features for more sophisticated machine learning algorithms. The added complexity is partially negated by the use of *morphological normalization*, such as *lemmatization* or *stemming*. However, these additional preprocessing necessities do imply additional error due to the occurrence of erroneous normal forms, depending on the reliability of used implementations.

Besides linking entities between themselves, *coreference chains* extend through pronouns as well. Example:

"***President George W. Bush*** (Person) *is set to speak at the summit in November. He is expected to address the latest geopolitical questions in the wake of recent terror attacks*."

In this case, we would like to automatically infer that *He*, in fact, refers to *President George W. Bush*. Besides NERC, this requires detecting nominal mentions and pronouns Often times *chunking* is used as a preprocessing step for NERC, which turns out to be practically useful for coreference resolution as well. State of the art for coreference resolution is currently based on machine learning methods (Glavaš and Šnajder (2015)), even more so than widely used entity linking methods.

Named entity linking may supplement coreference resolution schemes by the means of base matches of named entity mentions. If done properly, coreference resolution can then be implemented by grouping named entities and mentions based on their knowledge base matches. This turns out to be practically feasible where relatively bigger knowledge bases, such as Wikipedia, are available (Hachey et al. (2011)), effectively reducing coreference resolution to a subtask of named entity linking.

As is the case with entity linking, highly inflective languages add additional complexity to coreference resolution. On the other hand, such complexity may help solving some subproblems, e.g., noun-pronoun gender ambiguity. The impact of inflection can be smoothed via morphological normalization.

3. Joint Learning

3.1. Conditional Independence

Previous chapters surveyed mostly pairwise named entity relation extraction techniques. Rather than treating the extracted information as a graph, those methods attempt to extract named entity relations as individual edges, with two exceptions.

One exception was ILP based joint inference, described in 2.2.3, which attempts to adjust the ultimately predicted entity-relation by adding and removing some of the edges (named entity relations), and even adding, relabeling, and potentially erasing the nodes (named entities). This approach, however, is still conditioned on the verdict of classifiers that extract relations under the assumption of conditional independence between them.

The other exception was Snowball, briefly described in 2.2.2. Considering the *seed* as a perfectly annotated set of named entities and relations, Snowball does produce patterns and attempts to match more named entities and relations – based on previously extracted ones. The main difference between Snowball and other introduced algorithms lies in the fact that Snowball, at a given time step, does actually use the *joint* features of all named entities and relations extracted up to that point, as opposed to localized lexical and morphosyntactic features derived from the surroundings of potential named entity and relation predictions. Let m_t^* be the exact match at step t, and m_t be potential matches, one of which eventually turned out to be m_t^*. Patterns generated at step t being based on matches at steps $1 \ldots (t-1)$ imply conditional modeling for m_t^*:

$$m_t^* = \arg\max_{m_t} p(m_t | m_{t-1}, m_{t-2}, \ldots, m_1)$$

Consider the case of three named entities, A, B, and C, and two types of pairwise relations, *kinship* and *friendship*. Where *kinship* over $\{A, B\}$ and $\{B, C\}$ would most

often imply *kinship* between A and C, it will usually not be the case if kinship is substituted with *friendship*, a non-transitive type of relation that requires explicit statements.

Such heavily contextual information sharing had motivated research in *structured prediction*, defined as follows:

Structured prediction as a means of solving tasks involving conditional dependencies first became prominent with the advent of probabilistic graphical models (Koller and Friedman, 2009). Graphical models are used to produce structured output by estimating the most likely sequences of partial structures, known as *states*, leading to the ultimate output structure.

Prediction using graphical models is done by applying variants of the Viterbi algorithm, described in Viterbi (1967), in order to solve the *argmax problem* and find the most likely structure for given data. The Viterbi algorithm is a dynamic programming algorithm used to estimate the most likely states at each step of construction. It models information sharing based on the structure of the underlying graphical model that is represented as a graph of either directed (hidden Markov models) or undirected (conditional random fields) edges, describing the flow of conditional dependence. When the Viterbi algorithm is used, the time complexity for predictive inference is $\mathcal{O}(TN^k)$, where T is the length of the input sequence, N the number of labels (types of components of the inferred structure), k being the *Markov order* of the structure.

Conditional random fields have been successfully applied to various tasks in natural language processing, such as shallow parsing (Sha and Pereira, 2003) and NERC (Glavaš et al., 2012). The possibility of picking virtually any factor graph to fit the specifics of the problem at hand (i.e., linear chains for sequence labeling) further adds to the expressiveness of graphical models, enabling application on a wide spectrum of domains.

A corresponding factor graph of a conditional random field leads to the following for the joint distribution of all labels, $\boldsymbol{y}$, conditioned on features $\boldsymbol{x}$ and model parameters $\boldsymbol{w}$:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}) \propto \prod_{i=1}^{N} \psi(y_i|\boldsymbol{x}, \boldsymbol{w}) \prod_{i=1}^{N-1} \psi(y_i, y_{i+1}|\boldsymbol{x}, \boldsymbol{w}) \tag{3.1}$$

$$\psi(\boldsymbol{y_s}|\boldsymbol{x}, \boldsymbol{w}) = exp(\boldsymbol{w}^T \phi(\boldsymbol{x}, \boldsymbol{y_s})) \tag{3.2}$$

The right side of 3.1 can be normalized by dividing it by the sum over all $y \in \mathcal{Y}$ in order to produce a proper probability mass function. Factors ψ (3.2), represent the basis for localized joint distributions, whereas ϕ is the function that maps data to feature

vectors.

The loss function of conditional random fields can be expressed in terms of log-likelihood geared towards maximizing the sum of probabilities for labeled instances, with a regularization term used to penalize overfitting model parameters $\boldsymbol{w}$. Training the model then amounts to minimizing the loss, which happens to be a non-linear optimization problem and therefore *intractable*. Training is usually done using stochastic gradient descent or Newtonian methods, such as Broyden, Fletcher, Goldfarb and Shanno's algorithm (Shanno, 1985).

The intractability of searching for optimal parameters poses a significant disadvantage in real world applications, where predictive systems often need to be re-trained on incoming data. As a disjointly-learned alternative, support vector machines can be trained in $\mathcal{O}(n^2)$, or even $\mathcal{O}(n)$ if linear kernels are used, via *sequential minimal optimization*, where n is the number of samples in the training set (Fan et al. (2008)). While tight bounds on asymptotic time complexity (convergence speed) for non-linear optimization algorithms are still being researched, their worst case complexities are essentially exponential for general graphs. For the special case of directed acyclic graphs, finding the optimal parameters can be done in polynomial time, albeit involving the application of the *forward-backward* algorithm (Murphy, 2012) individually to every training sample, leading to theoretically worse complexity in comparison with training algorithms for support vector machines. Training probabilistic graphical models to solve the same problem requires more time and processing power in practical environments (Daumé III et al., 2009). Another crucial disadvantage of graphical models is their tendency to overfit, because of the fact that features often display unwanted hidden correlations, which then influence the training algorithm into a specific optimal set of parameters – resulting in poor generalization of trained classifiers.

3.2. Learning to Search

So far we have discussed applying probabilistic graphical models (PGM) in order to implement joint learning. While PGMs are theoretically sound and practically viable structured prediction models when it comes to prediction alone, the sheer intractability of loss optimization presents a challenge for practical implementations. On the other hand, support vector machines are not as expressive and theoretically viable in the context of structured prediction, but they do compensate for such a disadvantage by being

relatively easier to train, due to their simpler loss functions. An ideal model would be both efficiently trainable and expressive while retaining generalization potential.

Daumé III et al. (2009) introduce a novel approach to structured prediction, based on state space search. The core idea is to observe structured prediction in terms of sequences of steps required to assemble predicted structures. Such steps are treated as *states* and the function used to guide search is termed *policy*, owing its name to a similar concept in the field of reinforcement learning.

Definition 1 (Daumé 2006)
A policy h is a distribution over actions conditioned on an input x and state s.

Note that policy is not a probability distribution. It's a function which, given the current input and state, provides the transition for advancing to the next state.

This approach reduces structured prediction to state-space classification. Training a functional structured prediction model requires training a multiclass classifier over state-input pairs.

3.2.1. Training

Policies are trained iteratively. Each iteration considers its policy to be the best one so far, producing weighted samples based on the input, states, and transitions encountered while assembling the predicted structure. The weighted examples are then used to train a new classifier in the next iteration, whose policy is interpolated with the previous policy into a new one, repeating the process. The first iteration often trains the initial classifier based on binary weight assignments – all training samples except ones leading to the preferred (labeled) outcome of structured prediction are assigned non-zero weights.

$$l_a^h = E_{(x,h,a)}[c_y] - \min_{a'} l_{a'}^h \tag{3.3}$$

Sample weights are based on relative expected losses of their respective actions, called *regrets* 3.3. A regret value l_a^h is based on the observed policy h, input x, action a, and expected conditional loss $E_{(x,h,a)}[c_y]$ over viable output structures y.

In case regret expectations happen to be too complex for practical computation, they can also be approximated (provided that a sufficiently rigorous efficient algorithm exists) or using Monte Carlo estimation (Daumé III et al. (2009)).

3.2.2. Initial Policy

The algorithm introduced thus far requires an initial policy to start the iterative improvement leading to a final one, used for prediction. Moreover, the initial policy should be *optimal*, meaning that it should deterministically produce minimum loss and with it the optimal sequence of actions leading to the assembly of the labeled structure $\hat{y}$, searching along the least regretful path. Formally, the optimal policy may be defined as follows:

$$h_t^*(x, y, \boldsymbol{c}) = \arg\min_{y_{t+1}} \min_{y_{t+2},\cdots,y_T} c_y \tag{3.4}$$

In equation 3.4, labels y_t refer to the partial structure assembled at step t, the policy h_t^* being the optimal one at the t-th construction step, with respect to input x, final output structure after T steps y and partial cost vector $\boldsymbol{c}$. The policy is conditioned on the expected loss of future decisions and the next step (relative to step t) in assembling the final prediction y. While the optimal policy serves as a perfect fit on the dataset, the primary goal of iterative improvement is to strive towards generalization, resulting in an interpolation of the initial policy and a learned one – the learned policy-based classifier might settle for bigger values of regret on the original dataset while being practically feasible for a wider variety of input.

3.2.3. SEARN

The algorithm, called SEARN, is a combination of the previously described ideas:

1. Start with an initial *optimal* policy h^*
2. Assemble a structure using the latest policy h
3. Produce state-input samples using the latest policy and map them to feature space
4. Compute *regret* weights for samples from (3)
5. Learn a new classifier (policy) h' from (4)
6. Let the latest policy h be an interpolation of itself and the newly trained policy h'
7. Jump to (2) and repeat

Algorithm: SEARN(X, h^*, ϕ Train, Convergence, AssemblySteps, Actions):

$h \leftarrow h^*$ **while** $\neg$ Convergence(h) **do**
  $S \leftarrow \emptyset$
  for each $x \in X$ **do**
    $s_{1:T_i} \leftarrow$ AssemblySteps(x, h) ▷ *partial structures until* s_{T_i}
    for $t = 1 \ldots T_i$ **do**
      $\boldsymbol{c} \leftarrow [\,]$
      let $\boldsymbol{\phi} = \phi(x, s_t)$ **in** ▷ *feature extraction*
      for each $a \in$ Actions(s_t) **do**
        let $l_a^h =$ computeRegret(s_t, a, h) ▷ *as described in 3.3*
        $\boldsymbol{c} \leftarrow \boldsymbol{c} + \left[l_a^h\right]$
      $S \leftarrow S \cup \{(\boldsymbol{\phi}, \boldsymbol{c})\}$
  $h' \leftarrow$ Train(S) $h \leftarrow \alpha h' + (1 - \alpha)h$ ▷ *policy interpolation*

The parameter α above is used as a measure of drift towards a newly learned policy, from the initial policy, which fits the training set. Values of α are often optimized using grid search and similar techniques because the parameter heavily influences the results on validation sets.

The convergence criterion is typically a number of iterations or a measure of difference between the last two learned policies. Once convergence is met, the latest policy is used to assemble a structure that is considered to be the prediction.

Even though any multiclass (or multiclass reducible) classifier may be used to implement policies, typical implementations rely on logistic regression and support vector machines (Daumé III et al. (2009)).

3.3. SEARN and Relation Extraction

Named entity relation extraction can be viewed from the perspective of structured prediction as the task of assembling an entity-relation graph based on textual data. Being able to use information from partially assembled graphs (incomplete structured predictions) as a predictive feature for classifying relations over individual named entity pairs should, theoretically, provide for better prediction results in the case where relations

are not strictly dependent on pairwise information.

In sequence labeling, *searching* means building a chain of labels, starting from either the beginning or end of the given text, ending up with a complete structured prediction at the opposite side. We propose a searching scheme similar to ones used in sequence labeling. Since named entity relation extraction results in general graphs, one is forced to choose an arbitrary starting point and direction for searching. In this thesis, we do so by assigning indices to named entities. Entity pairs are then observed as sorted tuples of corresponding entities' indices, meaning that a possible relation between entity 1 and entity 5 would be represented by $(1, 5)$. The tuples are then sorted by comparing their first indices and breaking ties with the second ones, in an ascending manner. A search policy then determines predicted relations in the order of entity pair indices, conditioning relations on previously predicted ones in the context of the direction of search.

Consider the following example:

"*While **Bob*** (Person) *and **Jerry*** (Person) *are both friends and colleagues at **ACME*** (Organization)*, the latter's sister **Anna*** (Person) *has been working for **ECMA*** (Organization)*, a company known to be investing in haptics jointly with the one they're working for.*"

We start by assigning indices to named entities in order of occurrence:

1. Bob
2. Jerry
3. ACME
4. Anna
5. ECMA

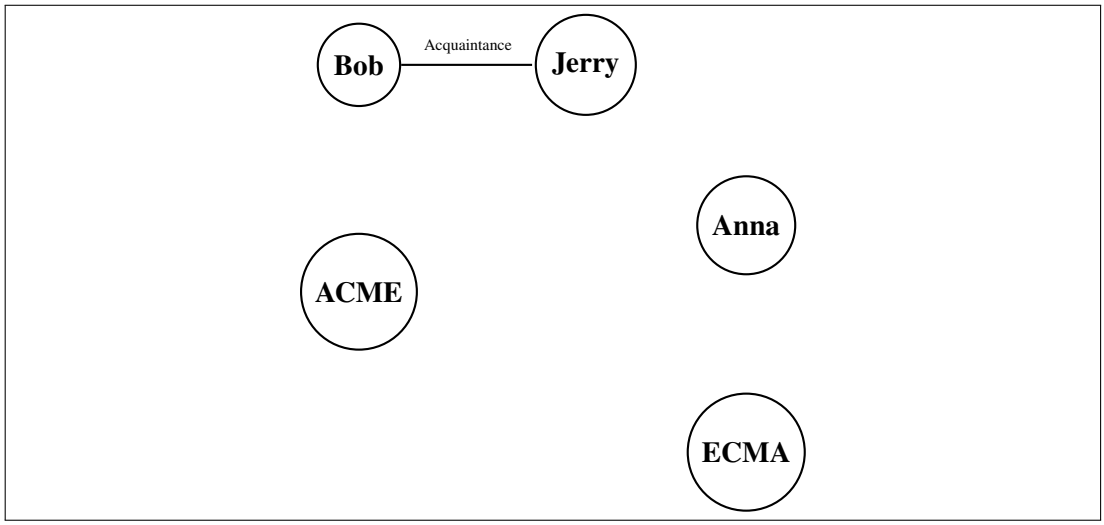
The potential relations are then sorted according to tuple indices:

1. (Bob, Jerry)
2. (Bob, ACME)

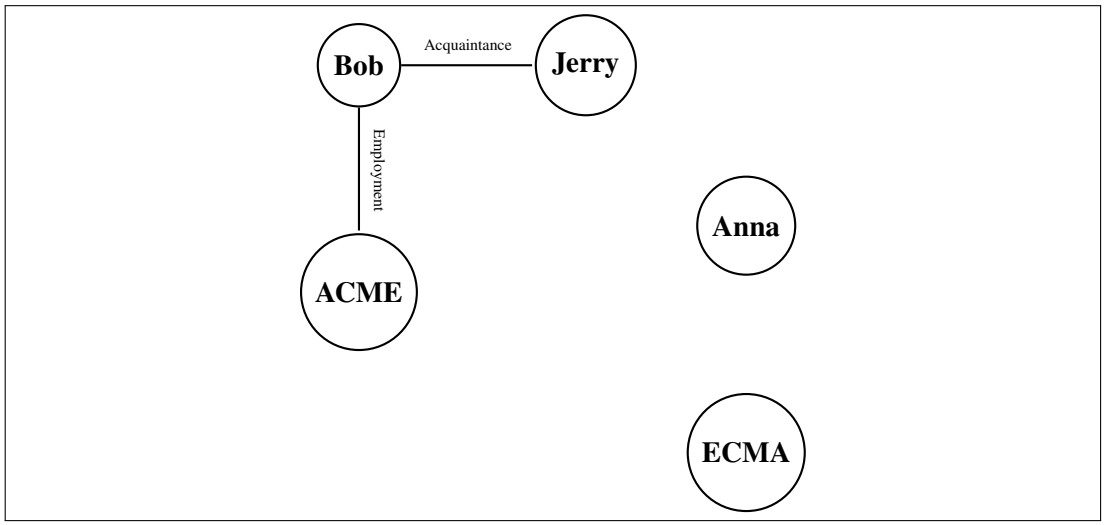
   ...

5. (Jerry, ACME)
6. ...
10. (Anna, ECMA)

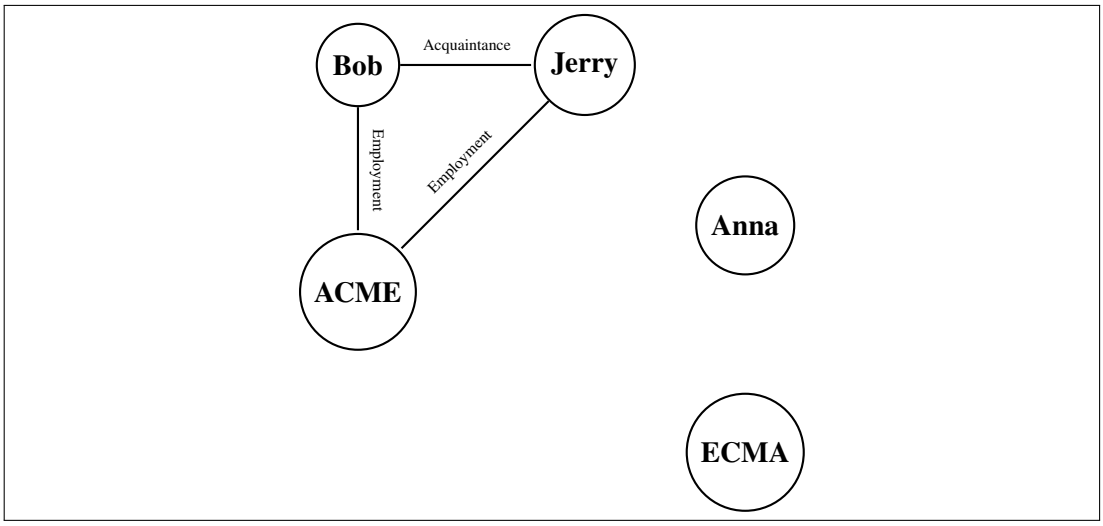
The search policy then attempts to assemble the structure starting in the order of entity pairs, which, given a simple entity-relation scheme, might result in structured prediction as described by the following partial structure graphs.



Step 1: Bob (*Person*) and Jerry (*Person*)

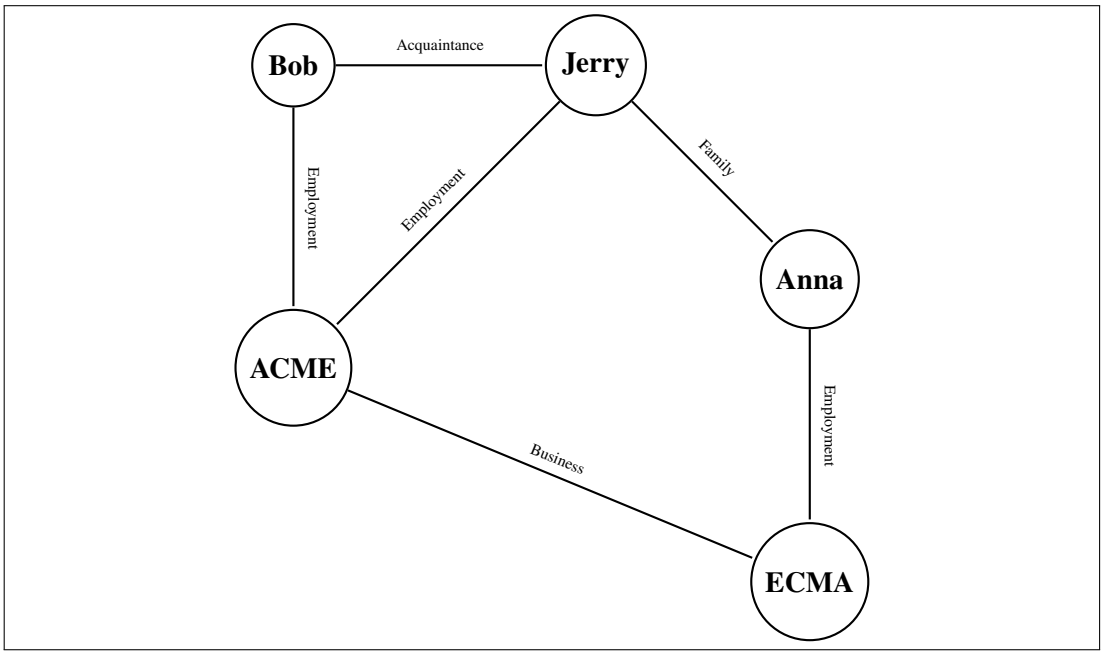


Step 2: Bob (*Person*) and ACME (*Organization*)



Step 3: Jerry (*Person*) and ACME (*Organization*)

• • •



Step 6: Anna (*Person*) and ECMA (*Organization*), final structure

Figure 3.1: Searching for a joint entity relation prediction

The choice of search direction is motivated by the fact that such an ordering allows for conditioning on previously observed nodes, whose incident edges have already

been predicted. In the given example, the relation between *Jerry* and *ACME* can be conditioned on not only *Jerry* and *Bob* being acquainted (as colleagues), but also on the relation between *Bob* and *ACME*, given that all of *Bob*'s relations have been predicted. However, the drawback of the approach is the fact that the relation between *Bob* and *Jerry* is not going to be directly inferred by the search policy. Such cases are handled during training – the policy being fine tuned to include a notion of shared information, regardless of the order of search, is a consequence of the iterative nature of the algorithm, where the deficiencies of one policy are being offset by the loss-based weights upon which its successor is trained. In theory, this means that the order in which single instance predictions are obtained should not significally affect the quality of joint prediction.

4. Data

In this thesis, we introduce a manually annotated collection of 591 newswire documents, written in Croatian. The NERC training set, provided by Glavaš et al. (2012), forms the basis of this collection, providing for manually annotated named entity occurrences. Our work, therefore, focuses on the annotation relations between named given entities. Using an internally developed tool and a single human worker, we have annotated a total of 4880 named entity relations in the course of estimated 80 working hours. We rely on parts of the complete dataset and cross-annotator validation metrics to estimate the quality of the dataset in terms of a theoretical upper bound for a named entity relation extraction algorithm, provided that no machine could surpass a well-trained human annotator.

4.1. Entity and Relation Types

4.1.1. Named Entities

The base NERC dataset (Glavaš et al., 2012) considers 11 distinct types of named entities: *Person*, *PersonPossessive*, *OrganizationAsLocation*, *LocationAsOrganization*, *Location*, *Ethnic*, *Percent*, *Money*, *Organization*, *Date*, and *Time*. We narrowed the types down to *Person*, *Organization* and *Location*, inspired by similar work, such as Culotta and Sorensen (2004) and Zelenko et al. (2003). Moreover, *PersonPossessive* was considered to be equivalent to *Person*, *OrganizationAsLocation* to *Location*, and *LocationAsOrganization* to *Organization*, resulting in a broader spectrum of named entities united under the three labels considered in this thesis.

.

4.1.2. Relations

We distinguish between four unique relation types:

Type	No. of occurrences
Person	6413
Location	4155
Organization	7301
Total	**17869**

Table 4.1: Named entity type occurrences, after mapping and filtering original types

1. *Influence*, of two *Person* type entities
2. *Affiliation*, of one *Person* and one *Organization* entity
3. *Association*, of two *Organization* type entities
4. *Location*, of a *Location* type entity and an either *Person* or *Organization* type entity.

We consider only relations that are explicitly documented within their respective documents. Any implicitly written relation or one which requires external knowledge of a human observer is considered to be a non-relation. As this dataset does not address the issues of named entity linking and coreference resolution, each and every occurrence of the named entity referring to the same actual entity is considered separately, its relations being based exclusively on linguistic semantics – this often means that, when a relation involving a named entity with multiple occurrences in the text is detected, it is set to be marked as a relation involving the latest occurrence of that named entity within the text thus far. For example:

„***Simon*** (Person) *instructed* ***Bob*** (Person) *to jump.* ***Simon*** (Person) *coerced* ***Bob*** (Person) *into leaving. He then decided that* ***Jerry*** (Person) *would be his new best friend.*"

In the given example, only the first mention of *Simon* would be considered as part of an interaction-based relation with the first mention of *Bob*, much like their respective second mentions would present an interaction-based relation. However, the second mention of *Simon* would also be considered to be in a friendship-type relation with *Jerry*, due to pronoun mentions, as well as coreference resolution binding them and the refered entity mention not being considered in this task.

Hypothetical and future statements are considered to be factual, meaning that our scheme ignores temporal information present within the text. Besides what was perceived as practical usefulness based on newswire text, relation types were chosen in a

way to be uniquely determined by types of related named entities, as formally defined in 2.2.1. This reduces the structured prediction problem to a simpler one – given text containing n named entities, find the optimal among $2^{\binom{n}{2}}$ subsets of possible edges.

Relations are considered throughout the whole document, meaning that a named entity occurring at the beginning of the first sentence of the document might present a relation with one occurring at the end of the last one. However, newswire text is typically written in a relatively simple form, meaning that two named entities forming a relation are usually found within the same sentence, or at the least in adjacent sentences.

Influence

Influence represents interaction semantics between the two given named entities of type *Person*. The form of interaction need not necessarily be *positive*, i.e., a partnership or friendship – *negative* relationships, such as murder or two persons being on opposite sides in a lawsuit are covered by *Influence* relations.

One exception to this general interpersonal relation would be authorship, not considered to fall under *Influence*. Suppose we have the following example:

"***Will Smith*** (Person) *reprised his role as the* ***Fresh Prince*** (Person)."

In the given sentence, *Will Smith* and *Fresh Prince* would not be considered as connected via an *Influence* relation, due to the former being considered as the author of the latter. However, in general, *Influence* does not depend on the fictitiousness of the named entities nor the relational semantics between them as expressed in the text. The key thing to note about *Influence* is the requirement of a strict *proof* written in text, rather than some hypothesis based on global knowledge.

Examples:

1. "***David Cameron*** (Person) *announced his resignation, to much dismay of his former deputy,* ***Nick Clegg*** (Person)."

   David Cameron and *Nick Clegg* used to be colleagues, indicating an interpersonal relationship. The relationship described as being in the past does not affect *Influence* (nor any other relation type for that matter), by the previously introduced convention.

2. "***Marta Scrassigna Robić*** (Person)*, wife of the late* ***Ivo*** (Person)*, sued his brothers* ***Rajko*** (Person) *and* ***Miroslav*** (Person)."

This sentence is an example in which all pairs of named entities are bound by *Influence* relations. *Marta Scrassigna Robić* and *Ivo* by marriage, *Ivo*, *Rajko*, and *Miroslav* by brotherhood, but also *Marta Scrassigna Robić* and *Rajko*, as well as *Marta Scrassigna Robić* and *Miroslav* – because of her lawsuit against the two.

Counterexamples:

1. " ***Sonja Kastl*** (Person)*'s superb choreography and costumes carefully curated by* ***Mirjana Zagorec*** (Person) *left a strong historical scent of times gone by.*"

   While *Sonja Kastl* and *Mirjana Zagorec* did work on the same play, there is no explicit indication of the two working together.

2. "***Farage*** (Person) *finished his speech by quoting* ***Peter Mandelson*** (Person)*, evoking an enthusiastic applause from his supporters*. "

   Farage did quote *Mandelson*. However, this does not imply that any sort of interpersonal interaction had occurred between the two.

Affiliation

An *Affiliation* is a relation between a *Person* and an *Organization* type named entity. Much like *Influence*, it can be thought of as an interaction between a person and an organization. Unlike *Influence*, it is restricted to *positive* relationships only. Consider the following example:

" ***Hulk Hogan*** (Person) *successfully sued* ***Gawker*** (Organization) *for defamation in one of the most devastating lawsuits in recent times*."

In spite of there being an interaction between *Hulk Hogan* and *Gawker* in the sentence above, it is not considered to be an *Affiliation* because of the negative nature of the relationship between the two – a lawsuit where subjects have stood on opposite sides.

Non-binding *weak* relationships do not count as *Affiliations*. An example would be a *Person* being interviewed by an *Organization* – while there is some, possibly positive relationship between the two, it is unclear whether the event implies a relationship of significance between the two, thus being deemed useless for practical use. Relations labeled as *Affiliation* denote *cooperation*, rather than *interaction*.

Examples:

1. "*Unlike his predecessor,* ***Thomas S. Monson*** (Person)*, the leader of* ***The Church of Jesus Christ of Latter-day Saints*** (Organization) *has regularly declined to be interviewed.*"

   This is a textbook example of an *Affiliation*, where a *Person* is explicitly stated to be the leader of an *Organization*.

2. "***Real Madrid*** (Organization) *had managed to best their finest opponents.* ***Gareth Bale*** (Person) *turned out to be a crucial asset in the dreaded match against the Catalans*."

   While a bit more implicit than the previous example, this sentence expresses some sort of positively geared relationship between *Gareth Bale* and *Real Madrid*, which checks out as an *Affiliation*.

Counterexamples:

1. "*The* ***Croatian State Archives*** (Organization) *are to release declassified from the time of the presidency of* ***Franjo Tuđman*** (Person)."

   The *Croatian State Archives* is handling information related to *Franjo Tuđman*. This obviously does not imply any connection between the two that cooperatively binds them and thus represents an *Affiliation*.

2. "***Gibbons*** (Person) *took the case to the* ***Supreme Court*** (Organization) *without second thought*."

   Gibbons invoking the *Supreme Court* does not imply any sort of positive relationship between the two are not said to collaborate or associate by any means as written in the text.

Association

Association is analogous to *Affiliation*, albeit the former being a relation over pairs of *Organization* entities. Two organizations are considered to form an *Association* if they are positively interacting or related in any significant way that is documented within the text of their occurrence. Likewise, conflicts between organization are not considered to be *Associations*.

The main exception to the rule is the case of multiple organizations being viewed as members of a parent organization. In this case, if the parent organization is referred to in form of a named entity, child organizations are considered to be in an

Association with the parent organization, but not between each other, unless explicitly stated in the text. An example would be listing k permanent members of the *United Nations Security Council*. In that case, members of the council would amount to k *Associations*, one for each one of them and the *United Nations Security Council*, as opposed to $\binom{k+1}{2}$ *Associations* (one for each pair of organizations, including the parent one). However, suppose two members of the are seen as associates, i.e., the *United States* maintaining a joint missile shield with *France*. Were that the case, the *United States* and *France* would present an *Association* regardless of them being associated with the parent organization. As for the case in which the parent organization is not denoted by a named entity, child organizations are not to be bound by *Association* type relations through their membership, because of the vagueness of child organizations' intra-organizational relationships.

Sporting events pitting two participants against each other (e.g., a game of basketball), as a case of *friendly* conflict, are considered to bind participants by means of *Association*.

Examples:

1. "***Conservatives*** (Organization) *and* ***UKIP*** (Organization) *form coalition to run* ***Plymouth City Council*** (Organization)."

   An obvious example of *Association – Conservatives* and *UKIP* being politically aligned, as well as them both running the *Plymouth City Council* suggests 3 *Associations*.

2. "*The* ***Lakers'*** (Organization) *formidable rivals, the* ***Los Angeles Clippers*** (Organization) *are to challenge them in the first matchup of the season*."

   This example depicts the exceptional case of conflict as part of sport. The two basketball clubs (seen as *Organizations*) are bound by a sports event – a game of basketball.

Counterexamples:

1. "*Are the disillusioned* ***GOP*** (Organization) *voters ready to galvanize behind the* ***Libertarian Party*** (Organization) *nominee?*."

   While the sentence implies some sort of connection between the *GOP* and the *Libertarian Party*, its true nature is vague and based on the unnamed mention

of the *nominee*. This, as well the lack of definite indications of the relationship being positive leads to the conclusion that this pair of entities does not form an *Association*.

2. "*The **European Commission*** (i)*s set to vote on **Ukraine's*** (Organization) *fate tomorrow morning*."

   The relationship between the *European Commission* and *Ukraine* is not clearly stated in the text, and therefore not an *Association*.

Location

Perhaps the relatively simplest of all relations types, *Location* binds a *Location* type named entity with *Person* or *Organization* type entities, carrying the semantics of location of non-*Location*, i.e., a *Person* or an *Organization* performing any sort of *direct* activity at a location. A direct activity is one that directly influences a location, and may include a person writing a travel log about a country, or an organization planning to open an office in a city. An *indirect* activity would imply consequences related to a given location while not involving a direct presence or influence of the named entity. For example:

"***Israeli*** (Organization) *territorial plans on the **Gaza Strip*** (Location) *contributed to the increase in the number of asylum seekers in **Europe*** (Location)."

Israel and the *Gaza Strip* are related, because of direct activity ("*territorial plans*"). On the other hand, *Israel* and *Europe* are not related, because the "*increase of the number of asylum seekers*" being an indirect activity, a *consequence*, rather than direct *Israeli* operation located or related to *Europe*.

Relative location, unless explicitly specified as *nearby*, either literally or using a distance metric does not fall into the definition of *Location*. As an example, the *Foreign Legion* deploying troops west of *China* would be considered underspecified, while the *Foreign Legion* deploying troops in front of the *Pyrenees* would be considered as specified enough to be a *Location* type relation.

Examples:

1. "*The border splits **Kashmir*** (Location) *into two parts. **India*** (Organization) *controls two thirds, while the remaining third is under the jurisdiction of **Pakistan*** (Organization)."

In this example, both *India* and *Pakistan* form *Location* relations with *Kashmir*, as indicated by their immediate geopolitical influence in the region.

2. "***Russia*** (Organization) *ought to keep vetoing the resolution as long as the* ***Azov Battalion*** (Organization) *is active in* ***Ukraine*** (Location)."

   This example displays a direct activity of the *Azov Battalion* in *Ukraine*, which is classified as a *Location* relation. However, the sentence holds no implication of *Russia* directly influencing or being active in *Ukraine*, meaning there is no relation between *Russia* and *Ukraine*.

Counterexamples:

1. "***VP Joe Biden*** (Person) *stated that peace in* ***Syria*** (Location) *is not possible under the current regime*."

   The *VP*'s statement on the situation in *Syria* does not connect him to the location in the sense of a *Location* relation, because there is no direct activity, influence or presence linking him to the *Location* type named entity.

2. "*The* ***United States*** (Organization) *invasion of the canal country resulted in moderate economic damage in* ***Nicaragua*** (Location)."

   While the *United States* did consequently influence the situation in *Nicaragua*, such influence is not considered to be direct, according to the previously introduced conventions, and therefore not considered to be a *Location* relation.

4.2. Quality

4.2.1. Metrics

We evaluate the quality of the dataset using *Cohen's kappa* (Cohen, 1968) and prediction using $\boldsymbol{F}_1$ *scoring* (Murphy, 2012).

Cohen's kappa

The kappa is a widely used statistical metric for chance-corrected inter-annotator agreement between two annotators. Assuming binary classification, let p_{ij} be the probability of annotator A choosing class i, and annotator B choosing j, $i, j \in \{1, 2\}$. The probability of consensus can then be computed as $p_c = p_{11} + p_{22}$. A value of $p_c = 0$ indicates

total discord, while $p_c = 1$ indicates total consensus, meaning that the annotators' labels match for each instance in the dataset.

However, the annotators' agreement may potentially be inflated by randomness. In case annotators chose labels uniformly at random, expected agreement can be expressed in terms of guess probabilities. Let $p_{1A} = p_{11} + p_{12}$ and $p_{1B} = p_{11} + p_{21}$ be probabilities of choosing class 1 for entities A and B, respectively, $p_{2A} = p_{21} + p_{22}$ and $p_{2B} = p_{12} + p_{22}$ analogous probabilities for class 2. The probability of random agreement can then be computed as $p_r = p_{1A} \cdot p_{1B} + p_{2A} \cdot p_{2B}$.

Finally, a measure of chance-corrected agreement can be obtained as $p_c - p_r$, and we obtain Cohen's kappa by normalizing the expression to the unit interval:

$$\kappa = \frac{p_c - p_r}{1 - p_r}$$

A generalization of Cohen's kappa for use in k-class classification is straightforward, as increasing the number of labels only means introducing more probabilities of the forms p_{ij}, p_{iA}, and p_{iB}, by expanding the set of labels for both i and j from $\{1, 2\}$ to $\{1, \ldots, k\}$.

$\mathbf{F}_1$ score

A standard measure of quality in the field of supervised learning, the $\mathbf{F}_1$ score is defined in terms of three types of output for binary classifiers:

1. **true positive** (tp) or *hit*, where the prediction matches the actual label
2. **false positive** (fp) or *false alarm*, where the prediction is positive for a negative actual label
3. **false negative** (fn) or *missed detection*, where a negative prediction occurs for a positive actual label

Two standard metrics are expressed using tp, fp and fn: *precision*, $P = tp \,/\, (tp + fp)$, and *recall*, $R = tp \,/\, (tp + fn)$. Precision measures the number of retrieved positive instances (matches) as a percentage of positive guesses, while recall is the rate of retrieved positive instances, as a percentage of actual positive labels.

Since the relationship between precision and recall is a trade-off influenced by the used algorithm and its parameters, the $\mathbf{F}_1$ score presents a combination of those as their harmonic mean:

$$\mathbf{F}_1 = 2\frac{P \cdot R}{P + R} \tag{4.1}$$

4.2.2. Evaluation

We start by selecting a set of 100 documents containing the largest numbers of relations, according to the initial annotation. We then split the set into four equally sized subsets, assigning each subset to three validating human annotators.

Annotator	**Cohen's kappa**
$A1$	0.6583
$A2$	0.5989
$A3$	0.6001
$A4$	0.5993
Average	$\mathbf{0.6142}$

Table 4.2: Cohen's kappas of annotators against dataset, 60 documents each

This results in four distinct annotations for each document in the set. We then proceed by computing 4 values of Cohen's kappa, between each of the evaluation annotators' annotations and the baseline dataset, shown in table 4.2.

Although the values of Cohen's kappa weren't particularly high, they were deemed feasible for machine learning experiments performed as part of this thesis.

5. Features

5.1. Preparing the Dataset

We conduct a handful of transformations on the original dataset prior to feature extraction. Both SVM-based and SEARN-based models rely on all features described in this chapter. Non-numerical features are packed into one-hot encoded vectors, meaning that each such feature increases the size of the respective feature vector by the number of possible value assignments, as a sparse sub-vector in which the positions corresponding to the given assigned value of the feature are set to 1, the others being set to 0. One-hot encoding helps avoid bias caused by assigning arbitrary values to non-numerical features, leading to impractically skewed feature vectors.

The only difference between SEARN and SVM feature vectors is the inclusion of inter-relation features for the former. These features represent labels of named entity relations predicted prior the relation that is central to the given feature vector – covering (nearly) all labels of previously predicted relations involving the first (by index) of the central relation's named entities, as well as the labels linked to its predecessor, using Vowpal Wabbit (Langford et al., 2011).

1. Tokenize and split sentences, an internally developed rule-based tool, separating sentences and transforming each one of them into a sequence of tokens. Tokenization and sentence segmentation do not pose a particular problem in Croatian language, as even manually devised rule-based systems tend to achieve near-perfect performance.

2. Proceed by filtering semantically irrelevant tokens using a manually compiled list of stop words. However, for the purpose of both feature extraction and further processing, we store both filtered and unfiltered sequences of tokens.

3.a Compute sentences' POS tags from unfiltered token sequences, using a pre-trained Croatian model (Agić et al., 2013a)) for the HunPos tagger (Halácsy et al., 2007).

3.b Lemmatize tokens of filtered sequences, using a lexicon-based lemmatizer (Šnajder and Dalbelo Bašić, 2008).

4. Construct dependency trees from unfiltered tokens and their respective part-of-speech tags, using a pre-trained model (Agić et al., 2013b) for the MST parser (Goldberg and Elhadad, 2010).

5.2. Lexical Features

In spite of the conceptual simplicity of lexical features, they do tend to provide useful differentiating power to classifiers in practice. The list of lexical features used in our implementation follows below.

1. Up to 11 lemmas between named entities,
2. Up to 4 lemmas preceding to the first named entity,
3. Up to 4 lemmas succeeding the second named entity,
4. The distance between the two observed named entities including stop words, as a scalar feature,
5. The occurrence of punctuation in the text between the observed entities.

A limit of 11 lemmas between a pair of named entities was chosen based on word distance statistics. Over 90% of named entity pairs within the dataset tend to fall within windows of 13 lemmas, ignoring stop words. The number of 4 lemmas preceding the first and succeeding the last (by order of occurence) named entities was chosen empirically, as larger values haven't had a significant impact on classifier performance.

While the stop word distance feature serves as a measure of redundancy between two named entities (possibly affecting the likelihood of certain relations), punctuation patterns were chosen to distinguish whether the observed named entities occur within the same sentence of not, as a possible indicator of their relatedness.

5.3. Morphosyntactic Features

Our implementation relies on two major types of morphosyntactic features: *dependency trees* and *part-of-speech* tags. Part of speech tags group words of similar properties into grammatical categories, while dependency trees' edges and labels provide the

information of semantic dependency between various parts of sentences. The drawback of the used dependency grammar (Agić et al., 2013b) parsing only separate sentences was mitigated by introducing an artificial text-level root for each document, as the parent of the roots of all sentence-level dependency trees within that document. The idea of connecting sentence-level dependency trees was motivated by the possibility of two related named entities occurring in different sentences.

As is the case with token lemmas, our feature vector includes 13 positions designated for POS tags of filtered tokens occurring between the two (potentially) related named entities, as well as an additional 4 tags preceding and 4 succeeding the pair of named entities. In Croatian language, POS tags point to the grammatical roles of words within sentences – for example, verbs are usually used to express grammatical predicates, while nouns are more often reserved for subjects or objects.

5.3.1. Dependency Chains

Unlike POS tags and token lemmas, dependency trees open an entirely new perspective on the issue of feature design, in form of *dependency chains*. Dependency chains are paths connecting named entities within dependency trees. Such paths traverse other words within the entities' sentences of occurrence, suggesting a type of relationship between them based on the labels of the tree's edges. Motivated by the success of dependency chains of Bunescu and Mooney (2005) and Culotta and Sorensen (2004), we introduce three types of dependency chain features.

The first type of dependency chain features are dependency tree labels on the path between two named entities, representing an encoding of the dependency tree. The second type of features are tokens of words occurring along the path, including both lemmas of relevant words and stop words. The third type of features are POS tags of words occurring along the path, inspired by augmented dependency trees described by Culotta and Sorensen (2004).

6. Results

We prepare for evaluation of machine learning models by splitting the dataset into one sixth and five sixths, the former being the evaluation set and the latter the training set. We then perform grid search for the linear SVM-based liblinear (Fan et al., 2008) model, optimizing its regularization constant by selecting the one for which the 5-fold cross-validation results turned out to be the highest. Note that the SVM-based model uses all the previously mentioned features. While cross-validation and hyperparameter optimizaton was manually implemented for the liblinear linear-kernel SVM model, SEARN-based models relied on Vowpal Wabbit's parameter optimizaton schemes (Langford et al., 2011). SEARN-based models used in our work use logistic regression (Murphy, 2012) for policy modeling.

Model	**$\mathbf{F}_1$ Score %**
Rule-based	13.02
Liblinear	25.71
VW lexical	22.11
VW POS	26.04
VW dependency	**27.46**

Table 6.1: Evaluation results

The rule-based model consists of exact matching – if a certain token from the manually compiled list of tokens appears between two named entities, then those entities are considered to be in the corresponding relation. A specific list was compiled for each type of relation.

Models optimized on training and test sets are evaluated on the evaluation set and their optimized $\mathbf{F}_1$ scores reported in table 6.1, where *VW lexical* stands for a SEARN-based model using exclusively lexical and numerical features, *VW POS* for a SEARN-based model using lexical, numerical and part-of-speech features, while the best per-

forming model, *VW dependency* is a SEARN-based model that relies on lexical, numerical, part-of-speech, as well as dependency tree features.

7. Conclusion and Future Work

The steady increase in the amount of unstructured text written in natural languages spurred the interest of many practical information extraction tasks, such as named entity relation extraction. Previous attempts at solving named entity relation extraction were mostly through supervised learning frameworks, often involving kernel methods (Zelenko et al., 2003). The current state of the art models, such as tree kernels (Culotta and Sorensen, 2004) are yet to be beaten in terms of standard machine learning evaluation metrics. However, recent advances in structured prediction, such as search-based structured prediction described by Daumé III et al. (2009) and the corresponding implementation by Langford et al. (2011), as well as a growing trend of efficient implementations of existing machine learning models (Fan et al., 2008), open up new possible directions for attempts at solving the task. While the obtained results do not necessarily challenge their English kernel based counterparts, this thesis presents an unprecedented approach at solving named entity relation extraction in Croatian language using search-based structured prediction.

Further research may attempt to combine dependency tree kernel methods and SEARN by using them as a search policy in an attempt to beat the current state of the art techniques through the use of a more sophisticated meta-algorithm, as opposed to standard entity pair classification. Improving the quality of the dataset in terms of inter-annotator metrics may also produce better results. Finally, the sheer increase of the dataset could perhaps open room for connectivist approaches, such as convolutional neural networks, which may then be used as search policies.

BIBLIOGRAPHY

Ž. Agić, N. Ljubešić, and D. Merkler. Lemmatization and morphosyntactic tagging of croatian and serbian. In *4th Biennial International Workshop on Balto-Slavic Natural Language Processing (BSNLP 2013)*, 2013a.

Ž. Agić, D. Merkler, and D. Berović. Parsing croatian and serbian by using croatian dependency treebanks. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, 2013b.

E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

S. Bird. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

S. Brin. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*, pages 172–183. Springer, 1998.

R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics, 2005.

R. J. Byrd and Y. Ravin. *Identifying and extracting relations in text*. na, 1999.

J. Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics, 2004.

H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. Mcarthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz. The expanding digital universe: A forecast of worldwide information growth through. *Information and Data 2007*, pages 1–21, 2010.

G. Glavaš and J. Šnajder. Resolving entity coreference in croatian with a constrained mention-pair model. *BSNLP 2015*, page 17, 2015.

G. Glavaš, M. Karan, F. Šarić, J. Šnajder, J. Mijić, A. Šilić, and B. D. Bašić. Croner: A state-of-the-art named entity recognition and classification for croatian. In *Information Society 2012-Eighth Language Technologies Conference*, 2012.

Y. Goldberg and M. Elhadad. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics, 2010.

R. Grishman and B. Sundheim. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471, 1996.

B. Hachey, W. Radford, and J. R. Curran. Graph-based named entity linking with wikipedia. In *International Conference on Web Information Systems Engineering*, pages 213–226. Springer, 2011.

P. Halácsy, A. Kornai, and C. Oravecz. Hunpos: an open source trigram tagger. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 209–212. Association for Computational Linguistics, 2007.

X. Han and L. Sun. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 105–115. Association for Computational Linguistics, 2012.

D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

A. Krishnamurthy, C. EDU, H. Daumé III, and U. EDU. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*, 2015.

J. Langford, L. Li, and A. Strehl. Vowpal wabbit. *URL https://github.com/JohnLangford/vowpal wabbit/wiki*, 2011.

C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

A. McCallum and W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

D. Roth and W.-t. Yih. A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document, 2004.

A. Sasane and K. Svanberg. *Optimization*. Department of Mathematics, Royal Institute of Technology, Stockholm, 2014.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

D. F. Shanno. On Broyden-Fletcher-Goldfarb-Shanno method. *Journal of Optimization Theory and Applications*, 46(1):87–94, 1985.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.

J. Šnajder and B. Dalbelo Bašić. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing and Management*, 2008.

Združeni model za ekstrakciju relacija između imenovanih entiteta

Sažetak

Pojavljivanja imenovanih entiteta (npr., ljudi, organizacija i lokacija) u tekstu često tvore odnose ili relacije, kao što su to prijateljstva, poznanstva, utjecaj, suradnja i mnoge druge. Ekstrakcija relacija među imenovanim entitetima važan je zadatak, posebice s obzirom na trenutno dostupne količine teksta pisanog prirodnim jezikom. Najčešće izvedbe temelje se na algoritmima nadziranog učenja, koji zasebno klasificiraju svaki od parova imenovanih entiteta s ciljem zaključivanja relacije među njima. Takvi pristupi ne uzimaju u obzir strukturiranu prirodu relacija u obliku međurelacijskih odnosa i ograničenja prilikom učenja modela, što ponekad djelomično nadoknađuju upotpunjavanjem predikcija globalnom optimizacijom. Aktualna istraživanja u području strukturne predikcije temeljene na pretraživanju pružaju nove mogućnosti za rješavanje problema ekstrakcije relacija. Ovaj rad usmjeren je na jedan algoritam strukturne predikcije temeljene na pretraživanju, SEARN, prilagođen za zadatak ekstrakcije relacija među imenovanim entitetima. Uvodi se novi skup podataka za ekstrakciju relacija na hrvatskom jeziku. Opisuje se model temeljen na pravilima, model temeljen na strojevima potpornih vektora te model temeljen na SEARN-u. Kvaliteta spomenutih modela ocijenjena je na uvedenom skupu podataka, korištenjem $\mathbf{F}_1$ mjere.

Ključne riječi: združeno učenje, SEARN, ekstrakcija relacija, imenovani entitet, hrvatski, obrada prirodnog jezika

A Joint Model for Named Entity Relation Extraction

Abstract

Occurrences of named entities (i.e., people, organizations, locations, etc.) often appear within text in form of relations, such as acquaintance, influence, collaboration, and various others. Named entity relation extraction is a task of great importance, especially considering the sheer scale of presently availabe text written in natural language. Typical implementations are based on supervised learning algorithms, classifying each pair of named entities individually, in order to determine the relation between the two. However, such approaches do not take into account the structural nature of relations in the form of intra-relational influences and constraints at training time, sometimes compensating for the resulting drawbacks by post-processing outputs via global loss optimization. Recent research in search-based structured prediction opens new venues for attempting to solve relation extraction. This thesis focuses of a search-based structured prediction algorithm, SEARN, adapted to the task of named entity relation extraction in Croatian language. We introduce a novel dataset for relation extraction in Croatian. We then proceed by describing and implementing a rule-based predictor, a configuration of support vector machines, and a SEARN-based model for named entity relation extraction. The models are then compared on the produced dataset, their performance being evaluated in terms of $\mathbf{F}_1$ scores and related metrics.

Keywords: joint learning, SEARN, relation extraction, named entity, Croatian, natural language processing