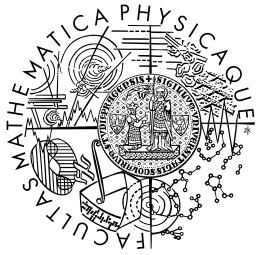

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Richard Ejem

Relation extraction in police records

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	doc. Ing. Zdeněk Žabokrtský, Ph.D.
Study programme:	Computer Science
Study branch:	Computational Linguistics

Prague 2017

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 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

Title: Relation extraction in police records

Author: Richard Ejem

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This work describes a problem of relation extraction between named entities on the sentence level, assuming that the named entities are already tagged in the text, on the domain of police reports written by the Anti-drug Department of the Police of the Czech Republic.

We have used various methods of machine learning in combination with tree kernel functions and methods based on sentence syntax rules.

None of the used methods had satisfying results on the data provided by the Police of the Czech Republic. Following analysis showed that tagging of the relations in the data was missing many relations, which were obvious to a human reader. That was found to be the reason why the supervised machine learning was not successful.

Later in this work we present several rules for recognizing relations which we have identified manually.

Findings in this work may be helpful for future research of processing these police reports.

Keywords: relation extraction, machine learning, natural language processing, Police of the Czech Republic, tree kernel

I dedicate this work to my beloved family and friends, who have been supporting me during all my studies.

Special thanks belong to my supervisor, Ing. Zdeněk Žabokrtský, Ph.D., for suggesting the topic of this work and his advices during the elaboration.

I also thank the Anti-drug Department of the Police of the Czech Republic for providing me with the data needed for this work and consultations about how the data are used.

Contents

1. Introduction

The task of extracting named entities and relations between them is an interesting NLP problem. As for many other NLP tasks, there are two approaches for this task: a domain–specific model or a general method to find and understand relations defined in natural language texts.

There are many practical applications of relation extraction. For example, search engines may provide better results by ranking found documents by relevant relations between keywords in a query. Another application is extracting some common information over entities, for example birth dates, relatives etc. Another use of relation extraction is to build a graph–like map of somehow related entities from documents from a narrow domain, allowing one to investigate an entity and important facts about it, for example connections between criminal suspects.

We will focus on the last one of these usages and try to build a relation extraction tool designed for police reports domain. There are many entity types that the Anti-drug Department of the Police of the Czech Republic nowadays describe in the text by manually annotating named entities and finding relations between them. For example, they point out persons figuring in reports, how they are related, where they were at some time, what car they are using, what drugs they carry/sell/use etc.

The provided part of the corpus owned by the Police of the Czech Republic consists of over 66,000 documents with about 115,000 unique named entities and 170,000 binary relations.

The goal of this work is to inspect existing relation extraction methods and fine–tune them or develop our own method adjusted best for the needs of the police reports. The resulting method will be implemented into a user–friendly piece of software, stand-alone or part of an existing linguistic tool.

A good candidate for such tool is TextAn[1], which has already prepared user interface for manual working with relations of named entities and uses Nametag to automatically recognize named entities.

The desired tool should be capable of the following tasks:

1. extract named entities from a plain text,

2. bind these entities with corresponding ones from other reports,

3. extract relations between entities from the text.

The first of these tasks is already done by the *TextAn* tool, which can find the entities and mark their positions in the text. Our work in this thesis is to develop a relation extraction method to find typed relations between named entities.

As the relations are not typed the in training data, we can not retrieve any type–of–relation information and learn from it. This police database just holds a graph of persons, cars, phone numbers and other entities that are somehow connected together, to help policemen inspect connections among suspects and criminals.

[1] `https://github.com/PreXident/TextAn`

We should be able to decide whether a pair of marked entities from a text is in a relation or not with a reasonable ratio of correctly detected relations. We use the provided police corpus to learn how to detect these relations.

We briefly introduce several methods that are possible to be used for relation extraction in Chapter 2, some of which we later base our experiments on.

Chapter 3 describes the format police corpus we have used for experiments. We have developed an application that transforms the corpus into a `connl.er` format describing sentence structure and entity relations in a simple form and runs our experiments. The structure of the application is described in Chapter 4. We have also proposed there a Representational State Transfer Application Programming Interface (REST API) for an application designed to only perform the relation extraction task to separate it from entity recognition and other corpus processing.

Various commands of this application are referred from chapters 5 and 6 where we describe processing of the data and our experiments.

Chapter 6 describes the experiments we have performed from machine learning to simple rule based methods. However, poor results these experiments led us to doubts about the quality of the corpus. Later on, we have decided to review the tagging of the corpus and consult the police officers, which revealed serious drawbacks described in Chapter 7.

Finally, we have at least identified that some types of relations, e.g. relation between two persons, can be predicted only from the fact that the two entities are present in a single sentence. We describe these rules in Chapter 8.

2. State-of-the-art in Relation Extraction

Relation Extraction is used to extract knowledge from an unstructured text into structured databases.

One commonly known application of Relation Extraction is mining structured information for search engines, so that highly relevant information can be provided when searching for something that can be described as a named entity of a particular type. For example, searching using Google for any famous person, city etc. shows structured information about it - a short description, date of birth and death, family and more. This information is most likely retrieved from online texts. Figure 2.1 shows structured information provided by Google on the query *"Albert Einstein"*. Orr [2013] from Google research describes Relation Extraction as one of the most difficult NLP tasks that can be much helpful in structuring human knowledge and exploring world's information.

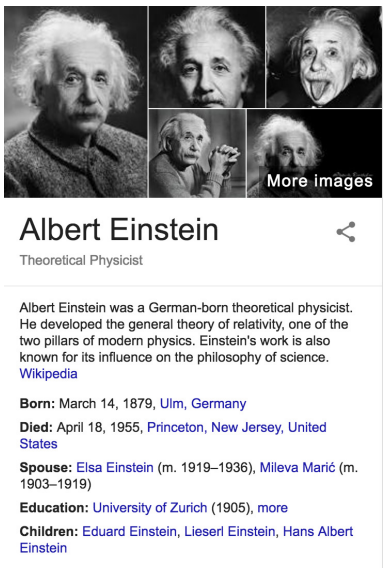

Figure 2.1: Structured information in Google search results

According to many resources about Relation Extraction found, for example Fundel et al. [2006], Chun et al. [2006] or Huang et al. [2004], Relation Extraction is widely investigated and used in bioinformatics to extract structured knowledge about gene diseases, protein interaction etc. from medical literature.

Current Relation Extraction methods are helpful, but far from 100% reliable. For example, Culotta and Sorensen [2004] achieved best 81.2% precision and 51.8% recall for relation detection (just a binary classification whether a given pair of entities is related or not) and 67.1% precision and 35.0% recall for relation classification.

It is not relevant to compare exact numbers from experiments on different corpora, as the success rate is strongly dependent on what kinds of relations are extracted and how complex and variable is the sentence structure expressing the relation.

Therefore we cannot compare results of this work with existing Relation Extraction systems. We develop our experiments for a very specific domain of the police corpus and we have found neither any other Relation Extraction software prepared to work on any similar domain and relation structure, nor any similarly structured corpus.

Bach and Badaskar [2007] reviews supervised and semi-supervised methods that are currently commonly used for Relation Extraction. Etzioni et al. [2008] classify KnowItAll and TextRunner as unsupervised. It is true that they do not need any labeled corpus. However, KnowItAll tag some relations in the corpus using defined linguistic rules at start. We think it is a bit disputable to consider such method unsupervised, because it actually gains initial "knowledge" from the rules.

In the following sections we introduce examples of selected supervised, semi–supervised and unsupervised (with rule-based initial tagging) Relation Extraction methods. We have decided to describe older methods from the materials we have found about relation extraction, because we see them as a base for various techniques used. More recent materials about relation extraction usually refer to these methods, trying to improve them.

The whole police corpus we have available for our experiments is labeled. Because of that, we estimate that we can achieve best results with supervised methods. Also, the nature of police reports does not change much in time, therefore we estimate that a model learned from the currently available corpus will not perform significantly worse over time.

We have decided to focus on supervised methods in this work. We do not suppose we would need discovering new relation expressions and adjusting the model using bootstrapping or similar techniques.

2.1 Supervised methods

Supervised methods are based on ML classification or regression, using a feature set $T(S)$ extracted from sentence $S = w_1, w_2...e_1, w_j, ...e_2, ...w_n$, using a classifier $f_R(T(S)) = 0|t_1..t_n$, where $t_1..t_n$ are relation types like *acts_in, is_located_in, created, ...* and *0* means no relation, or $f_R(T(S)) = 0|1$ for relation detection. Usually classifiers like SVM, perceptron etc. are used. Later in this work we describe our approach using a kNN binary classifier.

Supervised approaches are furthermore divided into feature based methods and kernel methods.

2.1.1 Feature based methods

Feature based methods can be used in situations in which a relation can be detected using a simple set of limited features, like a word form, suffix, POS tag and lemma of the entities and of neighboring words. However, relations are often expressed by language structures of variable size and complexity and therefore selecting a reasonable feature set for relation extraction is a difficult task.

One possible approach to reduce the problem of sentence structure complexity is to use kernel methods. Using them we can develop higher-dimensional feature spaces, using more information from the sentence's dependency tree structure.

Later in this work we show results of experiments with feature–based models on the police corpus, which were less successful than later kernel methods.

2.1.2 Bag of features kernel

Common approach of solving NLP problems is the so-called *bag of words* model. Basically, if we want to express similarity of two sentences in a number, we look at the sentences as unordered sets of words and count the number of words that are present in both sentences.

Usually, *stop–words* (frequent language words like junctions or modal verbs) are removed from the bag and only more informative words are kept, to make more relevant decision about semantical similarity of the sentence content.

On the other hand, when we need more to compare the sentences structurally rather than semantically, it may make sense to do the opposite - keep only frequent words that usually express sentence structure in the bag.

This approach can however be used more generally. The bag can be built not only from words, but various features of the words — lemmas, POS tags, constituent types etc.

2.1.3 Tree kernel

A tree kernel is designed to compute similarity between two dependency trees containing the examined pair of entities. Both Fundel et al. [2006] and Zelenko et al. [2003] use similar tree kernels. For two rooted subtrees T_1 and T_2, kernel function $K(T_1, T_2)$ is recursively calculating matching and similarity function of nodes in the subtree, starting in its root.

Matching function is defined as:

$$m(P_1, P_2) = \begin{cases} 1 \text{ when nodes are matchable} \\ 0 \text{ otherwise} \end{cases}$$

and similarity function in general as:

$$s(P_1, P_2) \in [0, \infty)$$

where higher number means more similar nodes. P_1 and P_2 stand for any node in subtree T_1 and T_2 respectively.

Simple similarity function may look like:

$$s(P_1, P_2) = \begin{cases} 1 \text{ if } P_1.lemma = P_2.lemma \\ 0 \text{ otherwise} \end{cases}$$

The kernel function K of the subtree is calculated starting with comparing features (lemma, POS tag, dependency relation type etc.) of two dependency tree nodes P_1 and P_2

- if they are not matchable, return 0

- if they are matchable, return $s(P_1, P_2) + \sum \lambda K(P_1.c, P_2.c)$ where $P_n.c$ are child nodes of P_n and λ is a "decay factor" to penalize nodes deeper in the tree.

According to Bach and Badaskar [2007], dependency–path tree kernels show to be best among all other kernel methods.

The biggest disadvantage of supervised learning is that it needs enough labeled data to train (it depends on the particular data and problem what does it mean "enough data"). The labeled corpus available for this work described in Section 3.1.1 seems to be large enough to us to try using supervised machine learning methods for the aim of this work.

2.2 Semi–supervised methods

2.2.1 DIPRE

Dual Iterative Pattern Relation Expansion (DIPRE), as described by Bach and Badaskar [2007], is a simple method of iteratively inducing patterns from a set of labeled examples and applying them to find more examples, based on matching longest common prefix and suffix string, developed to extract *author,book* relation from the web.

The word *dual* refers to duality between patterns and relations. With a set of "good" patterns, we can extract a set of tuples with both high precision and recall. Also, having a "good" set of tuples, we can determine a set of highly precise patterns.

DIPRE works as follows:

1. Start with a small sample of relation tuples R provided by the user.
2. Find all instances of R in a corpus.
3. Generate patterns based on the set of found occurrences. The patterns should be restricting, with as high precision as possible, because false positive instances would furthermore decrease precision when passing to following iterations.
4. Use the generated patterns to find another tuples in the corpus and add them to R.
5. If R is large enough, finish, otherwise return to step 2.

The big problem of this method is that it can get very unstable due to even small amount of bogus tuples in the initial set R or added to R in early iterations. More and more tuples are false identified as a relation in following iterations.

Brin [1999] describes in detail usage of DIPRE for the extraction of *author, book* pairs from the World wide web. An interesting part of described implementation is that they include Uniform Resource Locator (URL) prefixes in the patterns to divide the identified patterns into groups by a website or its section in which they were identified. It helps to solve the issue of generality — it is difficult to generate a general pattern fitting any website with high precision, but it is much easier to create specific patterns for a domain or website section in which the pattern was identified.

2.2.2 Snowball

Snowball is developed to extract *organization,location* relation (location of organization's headquarters). It is based on the same learning cycle as DIPRE with several improvements:

- It uses normalized term frequency instead of exact matching.
- It uses named-entity tags in the patterns, so the the entity type is also recognized and used for matching.
- According to Agichtein and Gravano [2000], the best improvement of Snowball is evaluating confidence of extracted patterns, which is calculated for pattern P as

$$Conf(P) = \frac{P.positive}{P.positive + P.negative}$$

True positive instances can be clearly recognized at least for *(organization, location)* tuples that were already recognized as positive by previous patterns.

The problem lies in recognizing negative (or false positive) results found during semi–supervised learning. Snowball uses the advantage of the specific entity types *(organization, location)*: A company should have only one headquarters, so if the algorithm has already recognized location for an organization with enough confidence, tuples extracted by new patterns that assign the same organization's headquarters to a different location are recognized as negative.

Having the confidence of the patterns computed, Snowball gives low weight to low confidence patterns. Tuples extracted using these patterns are discarded unless they are supported with more selective patterns.

2.3 Unsupervised methods

2.3.1 KnowItAll

Unlike DIPRE and Snowball, KnowItAll is a domain independent large–scale system deriving domain–specific rules from generic patterns and using pointwise mutual information to compute probabilities of relations between given two entities. However, types of detected relations still must be entered by human. KnowItAll was introduced by Etzioni et al. [2004]. Its engine is composed of these modules:

1. **Extractor** handles extracting facts from a text using instances of generic extraction rule templates. An example of a generic template can be *Noun Phrase (NP), such as List of Noun Phrases (NPList)*, which extracts relations of type *is instance of*, so from a text *cities such as Prague, Paris or Berlin* it can tell that Prague, Paris and Berlin probably belongs to the class *city*.
2. **Search Engine Interface** queries public search engines like Google or Alta Vista with queries based on extraction rules, like *cities such as*. Then, it runs the Extractor on these pages with a high probability of successfully finding new facts by applying these extraction rules.

3. **Assessor** assesses the likelihood that Extractor's result are correct using pointwise mutual information between extracted instances and multiple phrases associated with given entity type (i.e. cities) and combines them using *Naive Bayes Classifier*.

4. **Database** – KnowItAll uses a relational database storage for storing patterns, entities and metadata.

As we have mentioned at the beginning of this chapter, it is disputable if we can actually consider KnowItAll as truly unsupervised. Unsupervised methods should be able to extract some information from unlabeled data *without any prior knowledge of the data given to the machine learning method*, for example classify a set of entities into two or more groups based on some recognized patterns without "knowing" what these groups mean. However, the Extractor component uses a rule based tagger to identify some initial relations.

2.3.2 TextRunner

TextRunner is designed to overcome the issue of limitation to relation types entered by human. Self–supervised Learner automatically labels its own training data, trains a binary classifier to be used to extract more candidate relations. Using an Assessor, candidate relations are assigned probabilities and trustworthy ones are used for next passes.

Etzioni et al. [2008] introduces TextRunner and compares it to KnowItAll. In their experiments, TextRunner has about 33% better precision than KnowItAll. TextRunner engine consists of these modules:

1. **Single–Pass Extractor** uses simple part of speech tagger and a maximum entropy model to extract and chunk noun phrases and mark these that are most probably entities. It heuristically eliminates less significant phrases like prepositional phrases, adverbs etc., reducing the phrase to a normalized form of the relation.

2. **Self–Supervised Learner**, which, in contrast to KnowItAll and other algorithms, is capable of tagging an unlabeled corpus. It extracts tuples of the entities and a string denoting the relation $(e_i, r_{i,j}, e_j)$ and checks several heuristic constraints. If any of them fails, the tuple is tagged as a negative instance, otherwise it is tagged as a positive instance. Examples of used heuristics are:

   - Dependency chain between e_i and e_j exists and has a certain maximum length.
   - The path from e_i to e_j does not cross a clause boundary.
   - Neither e_i nor e_j consist only of a pronoun.

   When the Learner extracts positive and negative tuples, it maps them to a feature vector representation, which is used to train a Naive Bayes classifier.

3. **Redundancy–Based Assessor** uses counts of how many a relation was found in a normalized form and in any distinct form in different sentences to estimate the probability that $(e_i, r_{i,j}, e_j)$ is a correct relation.

   We can consider *TextRunner* to be truly unsupervised, because it replaces the rule based tagger used in KnowItAll with the Single–Pass Extractor, which uses only a part of speech tagger to initially label the corpus, but the relation extraction task itself is performed using maximum entropy model and heuristics only, therefore unsupervised.

3. Available resources

3.1 Available data

3.1.1 Police of the Czech Republic real dataset

We are collaborating with the Anti-drug Department of the Police of the Czech Republic, which currently has a set of over 66,000 text documents (see Table 3.1). The data were not processed uniformly — many of these documents have no entities or relations tagged, some have few entities with relations like two collaborating **persons**, **person** stopped in a **car**, contacting another **person** via **phone number**. Some documents also contain bigger number of entities, like lists of phone numbers to be monitored.

A type is assigned to each entity, so we can classify the relations by the types of the entities. Table 3.2 shows the number of entities of each type in the corpus (distinct entities, not entity mentions) and 3.3 shows the number of relation types in the corpus determined by entity types. We do not distinguish the order of entities in the relation so *Address – Person* is the same as *Person – Address*.

The police uses its own software for working with reports and tagging entities and relations in them. We do not have access to this software as it is a complicated closed–source system and they can not expose this software to external collaborants. They have prepared an XML export of these data, which is a starting point for our work.

As you can see in the snippet 3.1, the individual mentions of the entities in the text are not marked. The markup means only that they are mentioned in the document. We first need to locate entity references in the text to label the data for our experiments.

The fact that we do not have exact positions of the entities introduces another problem: we do not know which sentences actually represent a relation and which just contain entities related sentences, where the relation was expressed by another sentence or external knowledge of the annotator.

For example, the sentence *Novák Jan vystupuje pod přezdívkou SMITH* (*Novák Jan uses the nickname SMITH*) expresses a relation between a person's real name and nickname – so these entities express the same person. Later on, there is a sentence *Novotný volá Novákovi a ptá se, proč SMITH nedodal zboží* (*Novotný calls Novák and asks why SMITH did not deliver the goods*). *Novák* and *SMITH* are both mentioned in this sentence and marked as related entities. However, the relation can hardly be derived from this sentence.

These sentences may lead to problems with machine learning methods, as the learner is basically provided with some false positive instances with no linguistic clues of the relations. Also, when such sentences appear in evaluation sets, recall is falsely lower, because such sentences should and probably will not be detected by the relation tagger. Or, the relation tagger may learn patterns from such misleading sentences and wrongly evaluate similar sentences not expressing a relation as positive instances, lowering the precision.

Also, we cannot use this dataset outside the police department for security reasons. We needed to do all experiments with the corpus physically visiting

Documents	66,120
Sentences	168,331
Named entity occurrences[1]	723,225
Relations between entities in single sentence (positive instances)	117,476
Possible pairs of entities in single sentence (candidate instances)	3,455,216
Rate of positive instances to all entity pairs	3.4%

[1] Occurrences are not tagged in the corpus. We have counted the occurrences by tagging them ourselves as described in chapter 5.2.4.

Table 3.1: Police corpus characteristics

Entity type	Number of entities
Phone	36,846
Person	35,315
Address[1]	22,513
Car	9,650
Firm	5,633
Account[2]	247

[1] Address is actually misspelled as "Adress" in the whole corpus.
[2] Bank accounts

Table 3.2: Histogram of entity types

the police department. This has made the development and experiments more complicated.

Anonymization of the police corpus

This work contains anonymized examples of sentences from the real police corpus. All references to real entities – names, phone numbers etc. were replaced by random names and some facts (like amount and type of drugs) were changed to remove possible references to real cases.

Otherwise, the syntactic structure of these example sentences was kept as is. We present these anonymized sentences there in the original language (Czech) with translations to English. We tried to keep sentence structure in the English translations similar to original sentences. Therefore the English translations do not exactly respect word order of English in favor of pointing out original sentence structure in the corpus.

Also, some of the "sentences" are not correct grammatical sentences, mainly there is a missing verb. There are for example fragments of text recognized as a sentence by Treex[1] because it ends with a dot: "JAN NOVÁK, tel 777123456, bytem Politických vězňů 1, Praha" *(JAN NOVÁK, phone number 777123456,*

[1]Linguistic tree analysis tool developed by Institute of Formal and Applied Linguistics (Ústav formální a aplikované lingvistiky) (ÚFAL), `http://ufal.mff.cuni.cz/treex`

```xml
<?xml version="1.0" encoding="utf-8"?>
<Document Id="1080000504284">
<PlainText>
<![CDATA[
Osoba TEST Test, nar. 1.10.1983/1234, zaměstnán u firmy TEST
a.s., IČO 123456, jel dne 1.1.2008 v 20:00 s vozidlem
ŠKODA FAVORIT, bílá barva, RZ TTT1234. Při náhodném zastavení
hlídkou PČR bylo v zavazadlovém prostoru objeveno 5kg
OPL - kokain. Spolujezdcem byla osoba HOŘÍNEK Jakub
nar.7. května 1987.
]]>
</PlainText>
<Entities>
    <Entity id="1080000353709" type="Person">
        HOŘÍNEK JAKUB, nar. 19870507
    </Entity>
    <Entity id="1080000504286" type="Person">
        TEST TEST, nar. 19831001
    </Entity>
    <Entity id="1080000504294" type="Firm">
        TEST, IČ 123456
    </Entity>
    <Entity id="1080000504302" type="Car">
        TTT1234,ŠKODA ,BÍLÁ
    </Entity>
    <Entity id="1080000504312" type="UO">
        kokain, 5kg
    </Entity>
</Entities>
<Edges>
    <Edge parent="1080000504286" child="1080000353709"></Edge>
    <Edge parent="1080000504286" child="1080000504294"></Edge>
    <Edge parent="1080000504286" child="1080000504302"></Edge>
    <Edge parent="1080000504302" child="1080000504312"></Edge>
</Edges>
</Document>
```

Code Snippet 3.1: Example police report

Entity types	Positive instances
Person – Person	26,424
Address – Person	27,244
Person – Phone	18,968
Phone – Phone	12,661
Car – Person	12,793
Firm – Person	7,095
Address – Firm	4,415
Address – Car	2,772
Car – Firm	1,930
Firm – Phone	1,419
Firm – Firm	595
Address – Phone	447
Car – Car	258
Account – Person	209
Account – Firm	74
Car – Phone	67
Address – Address	47
Account – Account	46
Account – Address	8
Account – Phone	4
Car – Account	0

Table 3.3: Histogram of relation types

address of residence Politických vězňů 1, Praha). This does not matter much, because the information important for relation extraction (person, phone and address) are present here in a structure that could be also used in a sentence. For simplicity, we talk about such text fragments as sentences for the purpose of relation extraction.

3.1.2 Dataset from LDC

For the development of our system, we needed to find a corpus with marked positions of entities in text, which we would have available anywhere when developing the system. Unfortunately, entity relations are not tagged in common Czech corpora, such as the Czech national corpus and Prague dependency treebank.

We have found an English corpus in the the LDC catalog[2] published by Linguistic Data Consortium [2011] with catalog number LDC2011T08. This corpus contains tagged entities and relations. We have used it as another corpus for experiments with our relation extraction methods.

[2] `https://catalog.ldc.upenn.edu/`

Documents	646
Sentences	12,828
Named entity occurrences	23,700
Relations between entities in single sentence (positive instances)	2,473
Possible pairs of entities in single sentence (all candidate instances)	33,326
Rate of positive instances to all entity pairs	7.4%

Table 3.4: LDC corpus characteristics

This corpus has a wide range of tagged linguistic data, including dependency tree structures and named entity relations. It is stored in `XML` files. Each document is stored in 4 `XML` files:

1. *documentName*`.xml` contains the document's text parsed into paragraphs `<p>`, sentences `<s>` and sentence tokens (words and punctuation) `<w>`. It also contains simple token class tagging (`c` attribute of `<w>` tag). Classes are like "w" (word), "abbr" (abbreviation), "cd" (numeral), "." (dot) etc.

2. *documentName*`.ttt.xml` contains various tagging information. We use `w[l]` (lemma), `w[p]` (part-of-speech tag), `w[t]` from head word (dependency relation type to head node). It also includes all information from basic `.xml` file, so we do not even have to use the basic `.xml` file to build `.connl.er` files for our experiments.

3. *documentName*`.nrm.xml` contains a list of named entities `<ne>` in the corpus, with binding to sentences `ne[@sid]` and word range `ne[@fr]` to `ne[@to]` in which the entity appears, and relations `<rel>` between entities.

4. *documentName*`.dep.xml` contains information about dependency trees - relations between words and their head words are contained in `<dpg>` tags.

3.2 Current state of TextAn implementation

TextAn is a tool developed by students under supervision of the Institute of Formal and Applied Linguistics, containing user interface to tag entity occurrences, group entity occurrences as references to the same object, and relations between entities.

Automatic entity extraction is already implemented in *TextAn* using the *NameTag* tool. *TextAn* is also able to bind mentions of the same entity together.

We base this work on the fact that entity detection is solved by TextAn and express results of our experiments so that they can be used in a standalone application. Our system is designed to communicate with *TextAn* over REST API, adding automatic relation extraction ability to TextAn or similar tool. TextAn or another tool will send a text with marked entities to the API, receiving response containing pairs of entities that are related.

4. Developed application

We have developed an application to run our experiments that works with currently available resources. It analyzes raw text with tagged entities into a dependency tree structure and applies various rule based and machine learning methods to programatically suggest most likely relations between these entities. Then it evaluates and compares metrics Precision, Recall and F1-measure.

The technical solution of our application is consisting of these parts:

- **Treex** to parse raw sentences into word tokens with Part of speech (POS) tags etc. and analyze their dependency trees
- Our developed python application consisting of:
  - scripts for transforming the police corpus and LDC corpus into an unified `.connl.er` format,
  - scripts for extracting feature sets from a dependency tree with tagged entities,
  - scripts for evaluating various rule based and machine learning methods on extracted feature sets.

For our experiments with SVM models we have used the *scikit-learn* python library[1]. However, a k-nearest neighbor (kNN) classifier showed to achieve the best results with our kernel methods, so a custom implementation of a kNN classifier is used instead of *scikit-learn* for later experiments.

For actual integration of some ML method into TextAn, we suggest to use a very simple REST API interface described in the following chapter. It simplifies the whole problematics of relation extraction to tag relations between entities already tagged in plain text, keeping entity detection aside of relation extraction.

4.1 Interface

4.1.1 Request

The application expects a HTTP POST request, where POST body contains a plain text with marked mentions of entities like `<entity id="[0-9a-z]+"> entity text</entity>`.

```
Osoba <entity id="1080000504286">TEST Test</entity>,
nar. 1.10.1983/1234, zaměstnán u firmy
<entity id="1080000504294">TEST a.s.</entity>, IČO 123456,
jel dne 1.1.2008 v 20:00 s vozidlem
<entity id="1080000504302">ŠKODA FAVORIT</entity>,
bílá barva, RZ TTT1234. Při náhodném zastavení hlídkou
PČR bylo v zavazadlovém prostoru objeveno
<entity id="1080000504312">5kg OPL - kokain</entity>.
Spolujezdcem byla osoba HOŘÍNEK Jakub, nar.7. května 1987.
```

Code Snippet 4.1: Example HTTP API POST body

[1] `http://scikit-learn.org/`

4.1.2 Output

Output of the application is a set of relations between input entities, represented as a *JSON* array of pairs of entity identifiers.

```
[
    ["1080000504286", "1080000504294"],
    ["1080000504286", "1080000504302"],
    ["1080000504302", "1080000504312"]
]
```

Code Snippet 4.2: Example HTTP API response

4.2 Practical usage of the application

Until *TextAn* was developed, the police officers had to annotate the corpus manually. An annotator had to read the report document, manually assign mentioned entities (without mention position, just binding to the document) and relations between them.

With *TextAn*, entities can now be automatically extracted and also marked where they are mentioned in the text. The human annotator then just checks the results in *TextAn* GUI and fixes wrongly detected entities.

TextAn also contains GUI for marking the relations manually, but the annotator still must fill all the relations by himself. The tool developed in this work should help him with automatic detection of relations precise enough to be helpful — so that removing wrongly detected relations and adding not detected ones manually takes less time than entering all relations manually.

4.2.1 Objective of the experiments

The Anti-drug Department prefers recall over precision — better to give annotator more relation candidates rather than missing some. We take note of this, however we must also consider that low precision will be also a big problem. If the annotator was provided with a big number of instances and most of them were false positives, it would be more work to remove them compared to tagging the relations from scratch manually.

They were not able to tell precision and recall thresholds in numbers exactly, but we can estimate at least some lower bounds. Our rough estimate is that an absolute minimum for the application to be useful is precision is 0.5, because of the reason above - the annotator should not have to remove more relations than the number of actual relations.

Still, we want to achieve better precision than 0.5, let's say 0.6. When precision would be about 0.5, the usefulness of the application would be doubtful and it would be difficult to prove that it actually saves time and helps policemen to detect more relations, including a psychological aspect that the annotator will do the work all alone rather than to use a doubtfully helpful tool.

From our point of view, recall has no such lower bound – the bigger it is, the more time it saves to annotators, and even one correctly detected relation helps.

So, we can say that the aim of this work is to find a method with recall as high as possible, maintaining precision above 0.6.

4.2.2 Future work with the results

Currently (2017) it proves that the police actually does not use *TextAn* because it needs to be trained using lot more manually tagged data and they do not want to invest time into training it. They are currently having a contract aimed at a larger and better solution to be developed.

Regarding that and the fact that in this work we discovered that the current corpus is not tagged enough to perform any machine learning methods with good results better than simple rules, we have decided to include various scripts we have used for experiments in this work and identify problems in the corpus we were facing.

This work can be used as a report of methods that have been unsuccessfully tried and several simple rules we have identified that would be helpful for any future tagging system for police department.

5. Preparing the data

Our source data from the police corpus were stored in the format described in Section 3.1. We first needed to prepare an annotation mechanism that will expand these data into a format on which we can start machine learning experiments:

1. Extract entity positions from a raw text (will not be needed for the final module, which assumes this is done by TextAn or other tool for us).
2. Parse raw sentences into dependency trees with annotated lemmas and POS tags on word tokens.
3. Mark entities and relations between them in the dependency tree.

5.1 .er.connl format

We needed a format capable of representing a dependency tree. We have chosen a CoNNL format inspired by CoNNL-X [2006], from which we have removed the last two columns unused by our application and added two new columns for entity relations, calling this format *.er.connl*. This format is described in Attachment 3.

Sentences in *.er.connl* files are separated by one blank line. A single line containing document identifier in the format `@document_id` is prepended before each sentence, i.e. `@12345`.

It is helpful to identify the source document for example when random shuffling the data and separating a training/heldout/test set, because if test sentences would come from the same document as training sentences, overfitting[1] on particular documents could be hidden behind good precision/recall values. Overfitting actually does not occur on the test data, but may not be discovered when the test data are chosen improperly. For example, in this case, when both training and test data would be chosen from a smaller subset of documents related to the same case, with specific structures used only in them. We can expect that there may be some patterns repeating multiple times in a single document, but they may not be appearing in any other document. Splitting sentences from such documents into a training and a test set may also cause overfitting. We would be actually evaluating the model on the same set of documents as we trained it on.

Figure 5.1 shows the process of transforming the original corpus into *.er.connl*.

[1] The situation when a model gives much better results on the data it was trained or developed on than on any other data, where the reason is that the model fits strictly the particular training data rather than expressing general patterns of the problem it was designed to solve.

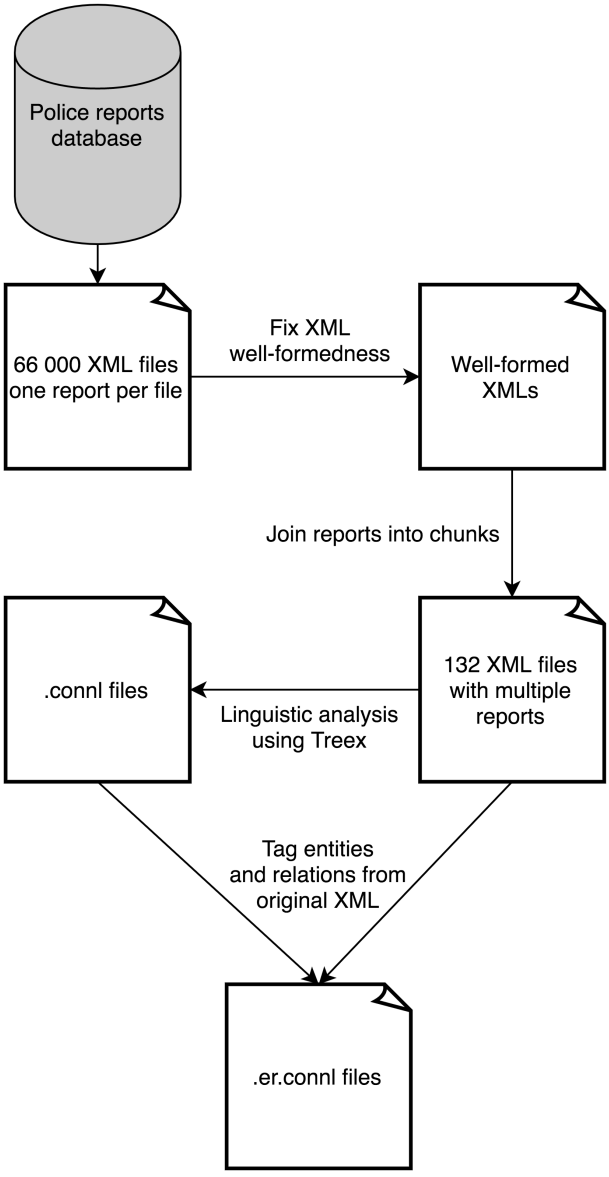

Figure 5.1: Transforming police corpus to .connl.er

5.2 Converting the police dataset to .er.connl

This section describes steps in which we processed corpus received in XML documents described in Section 3.1.1:

1. fix malformed XML documents,
2. join source XML documents into chunks,
3. do linguistic analysis on the raw texts from XML documents and store the results into the
   CoNNL-X format,
4. determine mentions of entities in the raw text and include them into the *CoNNL-X* output from the previous step, saving the result to `.er.connl` output file.

5.2.1 Fix malformed XML documents

As we have found, the corpus provided to us was exported into plain text from original RTF entries and wrapped in non-well-formed XML files by a script which we did not have control of. We were not able to parse these files directly using XML parsers because of the following problems:

1. Randomly occurring byte `0x0C` (Page Feed) which is not a valid XML character,
2. Unescaped & character,
3. UTF8 Byte-order mark (BOM)[2].

We have removed all `0C` bytes, all BOMs and escaped & into & using the script `fix_police_xml.py` included with this work, which did not remove or modify any important information from the document for our task. In our project, this fix can be run on sample police data using:

```
make police_fix_wellformed
```

5.2.2 Join source documents into chunks

We received the corpus in the form of more than 66 000 XML files in a single folder, which caused big difficulties in working with the corpus – loading and listing the directory was usually making the OS unresponsible. Each input XML file contained a single document (`<Document>` element from the corpus. Using the included script `generate_chunks.py`, we have combined these documents into 132 chunks (larger XML files) each consisting of up to 500 documents. The `<Document>` root elements from the original XML files were nested into a new root element `<Documents>`.

This can be done on sample police corpus in our project using this command:

```
make police_create_xml_chunks
```

[2]Byte-order mark, `http://unicode.org/faq/utf_bom.html`

5.2.3 Linguistic analysis

We have used Treex for linguistic analysis of the raw text, using the following chain of blocks wrapped in provided `police_xml_to_connl.sh` script:

1. Read and parse the data:
   (a) `Read::PoliceXml` – our custom module to read plain text from XML chunks of police corpus
   (b) `W2A::CS::Segment` – split plain text into sentences
   (c) `W2A::CS::Tokenize` – split sentences into tokens (words, punctuation etc.)
2. Annotate the tokens:
   (a) `W2A::CS::TagMorphoDiTa lemmatize=1` – add POS tags and lemmas to tokens
   (b) `W2A::CS::FixMorphoErrors` – Treex utility fixing common MorphoDiTa mistakes
3. Analyze dependency trees, also using the chain of scripts fixing common parser mistakes
   (a) `W2A::CS::ParseMSTAdapted`
   (b) `W2A::CS::FixAtreeAfterMcD`
   (c) `W2A::CS::FixIsMember`
   (d) `W2A::CS::FixPrepositionalCase`
   (e) `W2A::CS::FixReflexiveTantum`
   (f) `W2A::CS::FixReflexivePronouns`
4. `Write::CoNLLX` – output the result into CoNNL-X format[3]

We have chosen the CoNNL-X format because it contains information about the sentence's dependency tree and it is simple to parse and work with.

Using the included scripts

To execute this script on sample data, Treex has to be installed first:

1. Install Treex. See `https://ufal.mff.cuni.cz/treex/install.html` for instructions.
2. Copy `TreexPlugins/PoliceXml.pm` to:
   `<treex_dir>/lib/Treex/Block/Read/PoliceXml.pm`

Then, you can run the following command:

```
make police_connl
```

[3] `http://ilk.uvt.nl/conll/`

The `PoliceXml.pm` module adds the ability to read the police corpus in the XML format into Treex. It also adds dummy sentences in format *.documentid.* (the identifier of a document in original XML file), thanks to which we will be able to identify to which original XML documents the analyzed sentences belong and find entity–relation tagging in XML files for them.

5.2.4 Entity positions in police corpus

It can be seen from the source data sample 3.1 that there is no information which word or phrase represents which entity. Entities are just declared to be present in the document in the police corpus. We needed to pair these entities with their occurrences in the actual corpus. We have used the same simple approach as TextAn developers used for training TextAn entity tagging:

1. for each entity tag in the article, we extract its first word. Words are separated by spaces or commas. For example, the entity tag
   `<Entity id="1080000504312" type="UO">kokain, 5kg</Entity>`
   would be simplified to "kokain".

2. We match occurrences of the entities with forms and lemmas parsed by Treex. If the simplified entity matches any form or lemma from the text, it is considered that the entity is represented by this word.

Entities in the police corpus are mostly represented by proper nouns or other mostly unique words — names of people, places, car license plates etc. Therefore this approach is sufficient to identify most entities' mentions. This approach was also successfully used during the development of *TextAn*.

This conversion is done using `tag-entities.py`. This script extracts entities and relations from XML document chunks, finds corresponding tokens in the connl files using the heuristics mentioned above and produces a corpus in `.er.connl` format.

Using the included scripts

To perform this task on the prepared police connl and xml chunks, run the following command:

```
make police_er_connl
```

Finally, you can prepare sample police training, heldout and test set:

```
make police_split_sets
```

The whole transformation of police sample corpus into `.er.connl` sets can be done using these commands:

```
make police_prepare_sample
make police_split_sets
```

Please note that the police sample corpus is tiny, it is there just to demonstrate how the scripts work. No actual statistics can be produced from them, the actual numbers from the police corpus were calculated on the data that can not be included in this work.

5.3 Converting LDC dataset to .er.connl

The LDC dataset already includes tagging of syntactic dependency trees (all sentence tokens have a typed relation to the governing word/token) and entity relations. Unlike the police corpus, the LDC dataset is stored in well formed XML files, so the conversion into .er.connl is much easier than from the police corpus. We actually need only to transform the format, there is no need to perform linguistic analysis or fix formatting errors.

Each document and its metadata in LDC dataset is stored in 4 `xml` files:

- ***document_id*.xml** containing the document divided into paragraphs, sentences and word tokens,
- ***document_id*.ttt.xml** containing basic tagging: part–of–speech tags, lemmas, phrase types etc,
- ***document_id*.dep.xml** containing dependency trees data,
- ***document_id*.nrm.xml** containing named entities and relations between them.

Conversion from LDC dataset is done using `ldc2connler.py` script using the following syntax:

```
./ldc2connler.py input_folder output_folder
```

All the basic structural data contained in the main `xml` file can be found also in the `ttt.xml` file, so we do not read any data from the main `xml` file. This script does the conversion in the following steps:

1. It lists all triplets `ttt.xml dep.xml nrm.xml` of the same document id.
2. It parses these documents using the `ldc.LdcReader` class into a collection of sentences represented by `sentence.Sentence` class.
3. It writes sentences of each document into *output_folder*`/`*document_id*`.er.connl` using a function `connler.write_connler()`.

Finally, the documents are divided into train, heldout and test data using this command:

```
split_connler_sets.py input_folder output_folder --train=number
--heldout=number
```

The script automatically uses all remaining data for the test set. We use 1,000 sentences as heldout data, another 1,000 sentences as test data and remaining 10,818 sentences as training data for our experiments.

We have wrapped the whole process of preparing training, heldout and test set from LDC corpus into a make target in the root directory of our project:

```
make ldc_sets
```

This command puts training, heldout and test sets into `data/out/ldc/connler-sets`.

Figure 5.2 shows the process of transforming LDC corpus into .connl.er.

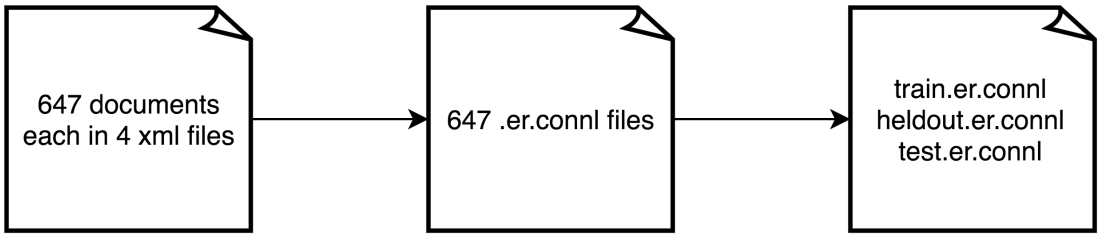

Figure 5.2: Transforming LDC corpus to .connl.er

6. Experiments

6.1 Classification of relations

For developing a machine learning model, we needed to extract pairs of entities from the corpus into featured instances.

We decided to extract all pairs of entities present in one sentence as instances, making a binary classification on them — entities are or are not related.

We have dropped relations between entities that are not found in the same sentence, as they are way more difficult to recognize by NLP. We have not found any existing reference to methods of relation extraction behind the scope of a single sentence.

6.2 Metrics and baseline

If we give all possible pairs of entities occurring in a single sentence to a classifier, we get four sets of results:

Result set	Short	description
True positives	TP	Pair of correctly recognized related entities
False positives	FP	Pair of unrelated entities misrecognized as related
False negatives	FN	Unrecognized pair of related entities
True negatives	TP	Pair of entities correctly recognized as not related

Table 6.1: Classification result sets

Based on sizes of these sets, we are measuring precision, recall and F_1-measure to rate and compare the results:

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

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

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

We have developed two baseline solutions:

- classify all candidate pairs of entities found in the same sentence as a relation
- a machine learning model using a very simple feature set – lemmas of words representing the entities are used as features

It is notable that results of both baseline solutions are low, not helpful for automatic detection of relations in the police corpus. Even the classifier with simple feature set has a precision much lower than 50%. It means that a human annotator would have to remove more edges than he would add without such automatic detection.

Baseline solution	Police			LDC		
	prec.	recall	f1	prec.	recall	f1
predict all pairs as relations	0.03	1.00	0.06	0.07	1.00	0.13
entity pair lemmas as features[1]	0.35	0.18	0.22	0.30	0.03	0.05

Table 6.2: Baseline values

6.3 Overview of methods tried

The following list shows and briefly describes which approaches we have tried, examining their efficiency (precision, recall and f1) and looking for the one having most helpful results for the task.

1. *Simple feature set* – a naive approach extracting a small feature vector, features extracted from parsed dependency tree,
2. *Dependency tree kernel* – feature vector consisting of tree kernel function from from Culotta and Sorensen [2004],
3. *Combining multiple kernel functions* – combining tree kernel with bag of words similarity and frequent words similarity.

6.4 Simple feature set

First, we designed a simple feature set extracted from the dependency trees, using the two nodes representing the pair of entities (first and second in the sentence) and their closest common ancestor node:

- The first and the second entity type,
- The first and the second dependency relation to parent node in dependency tree,
- The first and the second entity and their common ancestor's part of speech tag,
- The first and the second entity and their common ancestor's lemma,
- The distance from the first entity to common ancestor in the dependency tree,
- The distance from the second entity to common ancestor in the dependency tree.

The evaluation is divided into two steps. We first extract featured instances from the corpus (`.er.connl` trees with tagged entities and relations), then train and cross-validate a Support Vector Machine (SVM) and a kNN classifier on the training dataset.

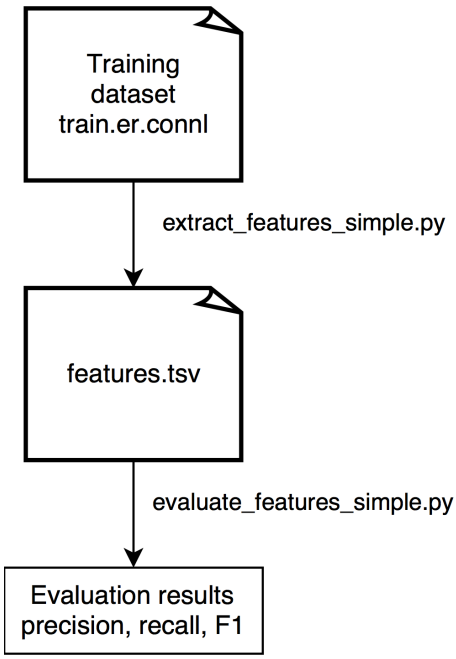

Figure 6.1: Evaluating machine learning methods on simple feature set

Extraction of features is done by `extract_features_simple.py` script. It reads all sentences from a `.er.connl` file and calculates feature values using `simplefeatures.extract_featured_instances()` function. Note that each sentence produces $\binom{\#entities}{2}$ candidate instances.

The resulting instances are saved into a `.tsv` file, in which each row corresponds to one instance. The first column determines if the instance is negative `0` (not a relation) or positive `1` (the two entities are related). Each of the following columns represent one feature.

Actual evaluation is done by `evaluate_features_simple.py` script. It uses a Python library named `sklearn` to evaluate both SVM and kNN machine learning methods.

The whole process is wrapped into a `make` target for both police and LDC corpus:

```
make evaluate_simple_features_police
make evaluate_simple_features_ldc
```

Attachments 1 and 2 show average precision, recall and f1 measure when we picked first n instances from the corpus, performing 5-fold cross-validation of the classifier. We evaluated SVM with linear kernel and kNN with several different numbers of neighbors.

We can see from the results that this approach gives quite good results with the LDC corpus, however poor results for the police corpus, for which it achieved maximum precision 0.31. That would not be helpful for a practical usage.

From that we can see that the simple features approach itself can work for relation extraction on some real data, however, not for the police corpus.

In the following chapters we will show if and how we can achieve better results using tree kernel methods and other more sophisticated approaches.

6.5 Dependency tree kernel function

We implemented a dependency tree kernel function introduced by Culotta and Sorensen [2004], calculating the gram matrix used by the classifiers, using our implementation of the contiguous tree kernel function based on K_1 *(contiguous kernel)* from that article.

In our scripts, the whole implementation of the tree kernel function can be found in `treekernel.compute.TKComputer` Python class.

The tree kernel function presented in the paper is defined as

$$K(T_1, T_2) = \begin{cases} 0 \text{ if } m(r_1, r_2) = 0 \\ s(r_1, r_2) + K_c(r_1[c], r_2[c]) \text{ otherwise} \end{cases}$$

This is implemented inside `TKComputer.k` method, which is an entry point to calculate tree kernel similarity function of two nodes from two sentences.

m is a matching function and s is a similarity function as described in Section 2.1.3.

Culotta and Sorensen [2004] define function $\phi(t_i)$ representing a feature vector of note t_i and consisting of two possibly overlapping subsets $\phi_m(t_i) \subseteq \phi(t_i)$ specifying features used for the matching function and $\phi_s(t_i) \subseteq \phi(t_i)$ specifying features for the similarity function.

Then, matching function is defined as:

$$m(t_i, t_j) = \begin{cases} 1 \text{ if } \phi_m(t_i) = \phi_m(t_j) \\ 0 \text{ otherwise} \end{cases}$$

and similarity function is defined as:

$$s(t_i, t_j) = \sum_{v_q \in \phi_s(t_i)} \sum_{v_r \in \phi_s(t_j)} C(v_q, v_r)$$

where $C(v_q, v_r)$ is a compatibility function between two feature values.

Unfortunately, there is no exact description of details of the kernel function used in Culotta and Sorensen [2004]. It remains unclear which features they used for matching function, how they implemented the compatibility function and what lambda value (the decay factor described in Chapter 5 of the paper) they used.

We had to figure out how to define these functions for our purposes from scratch. We developed these functions this way:

- the matching function returns 1 when entity type and part of speech tag of two nodes match. It is implemented in our `TKComputer.m` method.
- the similarity function returns 0.1 when only the part of speech tag matches, adds +0.4 when lemma matches and +0.4 when word form matches. It is implemented in our `TKComputer.s` method.

K_c is a similarity function over child nodes:

$$K_c(t_i[c], t_j[c]) = \sum_{a,b,l(a)=l(b)} \lambda^d(a) \lambda^d(b) K(t_i[a], t_j[b])$$

where $l(a)$ is length of sequence of indices a, $d(a) = a_n - a_1 + 1$ and $0 < \lambda < 1$ is a decay factor that penalizes matching sequences spread out within the child sequences. In our implementation, K_c is implemented as `TKComputer.k_c` method.

Culotta and Sorensen [2004] does not describe what lambda values did they use, so we used $\lambda = 0.5$ as a first guess for our experiments. As all the experiments were far from being successful, we did not even experiment with changing the lambda value much.

6.5.1 Evaluation of the tree kernel model

In our scripts, this evaluation on the LDC corpus is run using:

```
make extract_treekernel_ldc
make evaluate_treekernel_ldc
```

The process is also divided to kernel feature extraction and evaluation. The script `extract_features_tree_kernel.py` reads a set of sentences from a `.er.connl` file and writes the result into `features.tsv` and `features.tsv.test` in the folder `data/out/ldc/tree_kernel`.

The format of both these files is following:

- Each line corresponds to one instance (pair of entities).
- The first column is "0" (the pair is not related) or "1" (the pair is related).
- Each following column $[1..n]$ is a decimal number representing similarity to $(n-1)$-th instance from training data, calculated by the tree kernel function.

Also, we pruned such pairs of entities that have more than 40 nodes in the sentence dependency subtree, as calculating tree kernel function with every other instance takes very long time for experiments and we expect big amount of noise in such instances.

Another script `evaluate_tree_kernel.py` reads feature matrix produced by the previous script and runs an SVM classifier on it. It performs a 5-fold cross-validation by default, or evaluation on test set if `--test` argument is provided.

Table 6.3 shows average precision, recall and f1 measure when we picked the first 2000 instances from the corpus (182 positive, 1818 negative), performing 5-fold cross-validation and validation on test set of 1000 (62 positive, 938 negative) following instances on the SVM and kNN classifier using this tree kernel function. For kNN we have used 5 neighbors (5NN) as it gave the best results on the development data.

It is interesting to see the evaluation results of individual folds with the SVM classifier. The precision got low because of the last 3 folds, as there was only one true positive detected in fold 4.

We can also see that we were able to achieve better recall maintaining similar precision with the kNN classifier.

We also tried to apply this method to the police corpus. The results are presented in Table 6.5. Note that no positive instance was detected in cross-validation of SVM and testing kNN on evaluation set, resulting in precision and recall being 0.

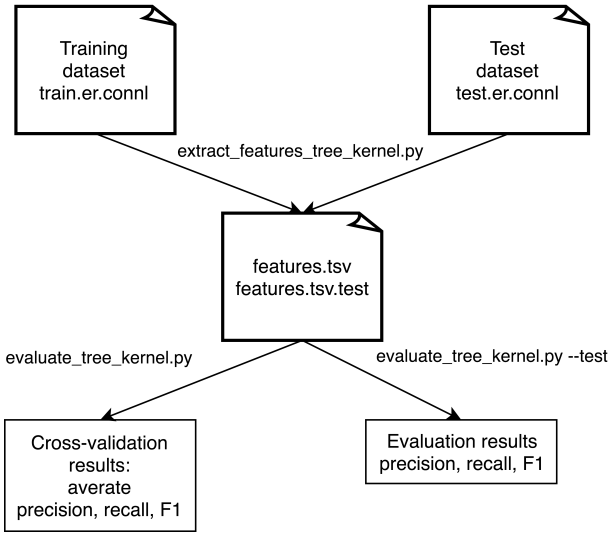

Figure 6.2: Evaluating SVM and KNN on tree kernels

Dataset	Method	Precision	Recall	F1
cross-validation	svm	0.44	0.12	0.19
test set	svm	0.33	0.01	0.02
cross-validation	knn	0.33	0.31	0.30
test set	knn	0.32	0.27	0.29

Table 6.3: Results on LDC corpus using tree kernel function

It may seem interesting that evaluation on test set with the SVM reached precision 0.44, which is significantly better than the result of kNN. But when we look to the evaluation results of test set (table 6.6), the classifier evaluated only 9 instances as positives, from which 4 were correct, which means the seemingly good result is not actually so significant.

We can see that the seemingly good precision is not very convincing in combination with a very low recall, for example there the precision was actually measured on only 9 retrieved instances.

Anyway, precision 0.44 would still not be useful for our application, and with recall 0.07 this model is obviously insufficient for practical use.

From these results we can estimate an uneven distribution of the relations in the police corpus, as one method (kNN) performed better during cross-validation and the other (SVM) on evaluation set.

6.5.2 Shuffled documents

We tried to select a random set of documents from the corpus when building the training and the test set. This should prevent overfitting potentially caused by using documents only from a certain part of the corpus, which may be the reason why two folds of cross-validation of tree kernel on LDC corpus gave no true positive instances. We can see that the evaluation results are now more similar for individual folds and the average precision, recall and f1 remains low.

If the result after shuffling was better, we would have to further investigate

Fold	Method	TP	TP	FP	FN
Fold 1	svm	20	337	0	43
Fold 2	svm	13	345	5	37
Fold 3	svm	0	374	1	25
Fold 4	svm	1	370	1	28
Fold 5	svm	0	380	1	19
Fold 1	knn	23	327	10	40
Fold 2	knn	21	322	28	29
Fold 3	knn	9	337	38	16
Fold 4	knn	3	355	16	26
Fold 5	knn	6	354	27	13

Table 6.4: Evaluation results, tree kernel, LDC corpus

Dataset	Method	Precision	Recall	F1
cross-validation	svm	0.00	0.00	0.00
test set	svm	0.44	0.07	0.13
cross-validation	knn	0.06	0.01	0.02
test set	knn	0.00	0.00	0.0

Table 6.5: Tree kernel, police corpus

Dataset	Method	TP	TP	FP	FN
test set	SVM	4	941	5	50
CV fold 1		0	334	0	66
CV fold 2		2	365	5	28
CV fold 3	KNN	0	379	0	21
CV fold 4		0	368	13	19
CV fold 5		0	359	0	41

Table 6.6: Evaluation results, tree kernel, police corpus

that the shuffle did not provide better results due to for example overfitting. However, no significant improvement just shows that these bad results were not caused by uneven data distribution.

Experiments with shuffled instances can be run by another Makefile target:

```
make extract_treekernel_ldc_shuffle
make evaluate_treekernel_ldc_shuffle
```

Dataset	Method	Precision	Recall	F1
cross-validation	svm	0.32	0.09	0.14
test set	svm	0.29	0.02	0.04
cross-validation	knn	0.29	0.18	0.22
test set	knn	0.29	0.40	0.33

Table 6.7: tree kernel, LDC, shuffled instances

Dataset	Method	TP	TN	FP	FN
Fold 1	svm	2	351	9	38
Fold 2	svm	4	364	5	27
Fold 3	svm	4	351	6	39
Fold 4	svm	2	364	7	27
Fold 5	svm	3	369	5	23
Fold 1	knn	5	353	7	35
Fold 2	knn	9	347	22	22
Fold 3	knn	9	340	17	34
Fold 4	knn	2	355	16	27
Fold 5	knn	6	358	16	20

Table 6.8: Evaluation results, tree kernel, LDC, shuffled

We can see from these results that evaluation on shuffled instances gives better results, however still not good enough for practical usage. Also kNN gives significantly better recall again.

Tables 6.9 and 6.10 show results of evaluation on shuffled instances on the police corpus. We can see an improvement in comparison with Table 6.5.

Dataset	Method	Precision	Recall	F1
cross-validation	svm	0.37	0.09	0.14
test set	svm	0.42	0.09	0.15
cross-validation	knn	0.49	0.05	0.08
test set	knn	0.25	0.06	0.09

Table 6.9: tree kernel, police, shuffled instances

Dataset	Method	TP	TN	FP	FN
Fold 1	svm	2	367	5	26
Fold 2	svm	4	372	5	19
Fold 3	svm	2	369	6	23
Fold 4	svm	1	375	7	17
Fold 5	svm	3	363	1	33
Fold 1	knn	1	370	2	27
Fold 2	knn	2	376	1	21
Fold 3	knn	1	373	2	24
Fold 4	knn	1	373	9	17
Fold 5	knn	1	364	0	35

Table 6.10: Evaluation results, tree kernel, police, shuffled

6.5.3 Training on more positive instances

Because the recall is still too low, we created another training sets containing a bigger ratio of positive instances than in original data. The test set was extracted from the corpus without adjusting positive instances rate in this experiment, because we only want to test if providing more positive instances for training improves the result. We've assumed that using unbalanced training set with more positive results will let the classifier to be trained to correctly classify more positive results – enhance recall, but possibly lower precision (it would be more difficult to properly recognize negative instances with less negative instances in training data). However, from Table 6.11 we can see that the precision is reduced too much to achieve reasonable recall, making this approach much less usable.

When performing cross-validation on such unbalanced training set, we have to properly calculate precision and recall regarding the unbalanced set. As demonstrated in Figure 6.3, unbalanced training set is actually distributed like a bigger training set, from which we removed a certain amount of negative instances, therefore the number of false positives and true negatives is lower during evaluation, true positives and false negatives remain unaffected.

When we look on formulas for precision and recall:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}$$

we can see that true negatives do not affect these functions and false positives affect only precision. To estimate the actual precision P on a corpus with original positive instances rate from cross-validation, we can use this formula to estimate the number of false positives on originally distributed corpus:

$$FP = FP_0 * \frac{i}{b}$$

where FP_0 is the number of false positives after evaluation on one fold, i is the rate of positive instances in unbalanced corpus and b is the rate of positive instances in original corpus.

Evaluations on the unbalanced training sets can be run by makefile targets:

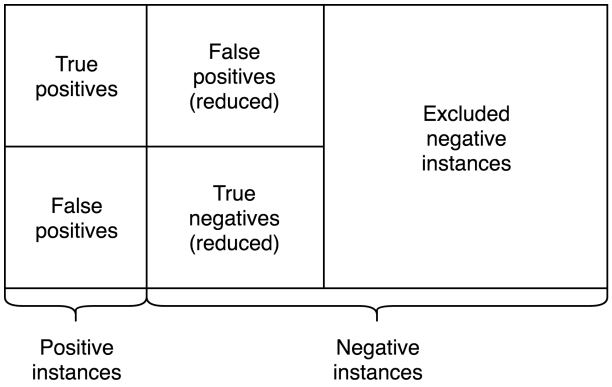

Figure 6.3: Evaluation results on unbalanced training data

evaluation	% positives[1]	precision	recall	f1
cross-validation	20	0.17	0.31	0.14
test set		0.30	0.36	0.33
cross-validation	30	0.24	0.49	0.11
test set		0.18	0.53	0.27
cross-validation	40	0.27	0.65	0.25
test set		0.15	0.63	0.24
cross-validation	50	0.41	0.83	0.36
test set		0.13	0.73	0.22

[1] how many % of all entity pairs (instances) used to train the classifier are actual relations (positive instances)

Table 6.11: Tree kernel, LDC, more positive instances

```
make extract_treekernel_ldc_p20
make extract_treekernel_ldc_p30
make extract_treekernel_ldc_p40
make extract_treekernel_ldc_p50
make evaluate_treekernel_ldc_p20
make evaluate_treekernel_ldc_p30
make evaluate_treekernel_ldc_p40
make evaluate_treekernel_ldc_p50
```

The police department prefers recall over precision — give annotator more relation candidates rather than missing some. We can see from the results that the expected result was achieved: we really improved recall, but precision falls below 0.25, which would not helpful for practical usage.

6.6 Combining multiple kernel methods

Because we have not found any relation extraction model good enough to be helpful for the police using the tree kernel method alone, we have tried to use

several other kernels and combine them together. Neither this did help us to find any significantly better model too, so we will just briefly describe the used methods and results.

Besides tree kernel, we have used two variants of a *bag of words* kernel and multiple simple kernels based on attributes of the pair of entities.

We have used kNN classifier with these kernels. Rough experimenting with the number of neighbors has shown to provide the best results (highest F1-measure) with 3 neighbors, so all results presented in this section were produced using 3NN model. As none of the results was even closely satisfying, we have not experimented with more configurations of the model.

6.6.1 Bag of words

For each pair of entities, we have found the first common parent word in syntax tree of the sentence and took all words from the subtree of this common parent as a bag of words.

When calculating two instances (pairs of entities P_1 and P_2), we have compared each word from the corresponding bag of words B_1 with each word from the bag of words of B_2. We have added 1 to the kernel for each matching word form, lemma or part of speech. The result is normalized by a product of number of words in the two bags B_1 and B_2. The kernel is defined as:

$$K_{bagOfWords}(B_1, B_2) = \frac{\sum\limits_{w_i.f=w_j.f} 1 + \sum\limits_{w_i.l=w_j.l} 1 + \sum\limits_{w_i.pos=w_j.pos} 1}{|B_1| * |B_2|}$$

assuming that $w_i \in B_1, w_j \in B_2$, $w.f$ is word form, $w.l$ is lemma and $w.pos$ is part of speech. Results can be seen in Table 6.12.

Corpus	TP	TP	FP	FN	prec.	recall	f1-measure
LDC	15	728	175	82	0.079	0.155	0.105
Police	3	921	25	51	0.107	0.056	0.073

Table 6.12: Bag-of-words kernel evaluation

6.6.2 Bag of frequent words

Another our kernel is based on frequent words. We have built a list of 1000 most frequent words FW in the corpus and defined the kernel K_{bofw} as a number of words present in bag B_1, B_2 and FW:

$$K_{bofw} = \sum_{w_i \in FW} \begin{cases} 1 \text{ if } w_i \in B_1 \wedge w_i n \in B_2 \\ 0 \text{ otherwise} \end{cases}$$

Results can be seen in Table 6.13. It seems that this kernel is useless for the LDC corpus, because it recognized only one positive instance out of 97. It is interesting that results on the police corpus were better than on the LDC corpus there. We assume it means that particular words may actually be a good clue

of a relation in the police corpus. However, both precision and recall is still insufficient for a useful model.

Corpus	TP	TP	FP	FN	prec.	recall	f1-measure
LDC	1	903	0	96	1.000	0.010	0.020
Police	12	841	105	42	0.103	0.222	0.140

Table 6.13: Bag-of-frequent-words kernel evaluation

6.6.3 Combinations

We have also experimented with combinations of the bag of words kernels presented above together with the tree kernel:

$$K_{combined}(B_1, B_2) = K_{tree}(B_1, B_2) + K_{bagOfWords}(B_1, B_2) + K_{bofw}(B_1, B_2)$$

Results can be seen in Table 6.14.

Corpus	TP	TP	FP	FN	prec.	recall	f1-measure
LDC	11	882	21	86	0.344	0.113	0.171
Police	17	897	49	37	0.258	0.315	0.283

Table 6.14: Evaluation of a combination of kernels

6.6.4 Using the attached scripts

Experiments with multiple kernels are done in two steps. First, matrices containing these kernels calculated on training and test data have to be generated using `extract_features_tree_kernel.py`. Second, a kNN classifier is applied on one or a combination of these matrices: `evaluate_kernels.py`. Example:

```
./extract_features_tree_kernel.py train.er.connl \
test.er.connl connler-sets/test.er.connl \
--outname=kernels/features.tsv \
--train=2000 --test=1000 --sim=tk --fwlist=wordfreq.dat

./evaluate_kernels.pyernels/features.tsv tk,bow,freq \
--neighbors=3
```

To clarify the arguments:

1. The argument *sim* accepts one of (bow,freq,tk) = bag of words, bag of frequent wods, tree kernel respectively.
2. The argument *fwlist* accepts a file containing list of words and number of occurences in the corpus in the format *numberOfOccurences theWord*, for example *12345 hello*, on each line.

6.7 Rule based approach

Experiments with machine learning did not gave us satisfying results. We have also noticed during our exploration of various relations appearing as false positive in evaluation of machine learning methods that many pairs of entities obviously tightly related, like two persons directly collaborating in drug business, are not tagged as a relation in the corpus.

This led us to perform another experiment. We have manually checked tagging of 500 randomly selected sentences of police corpus containing 1435 mentions of entities in total.

Based on sentences examined during that tagging, we have developed a set of rules for extracting relations. Because we were limited to work with the corpus only at the police department personally and for a limited time, we have focused only on extracting relations between persons, which are the most important for the police.

We have evaluated those rules on the small portion of the corpus where we have manually improved tagging and also on the whole corpus.

The rule based tagger is implemented in the set of scripts attached to this work as `tag_rule_based.py` script. It can be executed using

```
./tag_rule_based.py --rules="CLOSE_CONJ,PREDAT,JEDNA_O,\
V_KONTAKTU" input.er.connl output/confusion/matrix
```

where the first argument is an `.er.connl` file containing a portion of the tagged police corpus and the second argument is a folder where a list of true positive, false positive and false negative instances are stored for examination of instances on which the used rules do not work.

To simplify these experiments, we have preprocessed the corpus by another script `filter_instances.py`. This script removes all entities of other type than *Person*, then all sentences where less than two entities were left as they do not contain any possible relation.

The script receives two arguments, input ant output `.er.connl` file:

```
./filter_instances.py input.er.connl output.er.connl
```

Removing entity types other than *Person* is hard-coded in the script because we did not need any other filtered entity and relation types for our experiments.

Figure 6.4 shows the whole process of evaluating rule based methods on the police corpus in `.er.connl` format.

6.7.1 Relation extraction rules

We have defined the following four rules by examining about 500 positive *Person* – *Person* relations from the corpus. The rule tagger marks a pair of entities if at least one of the following rules matches. Each rule in this list is prefixed with an identifier of the rule, which is also used to define which rules should be applied by the tagging script.

1. **A**: Mentions of the entities are divided by at most 2 words and one of them is “a”.

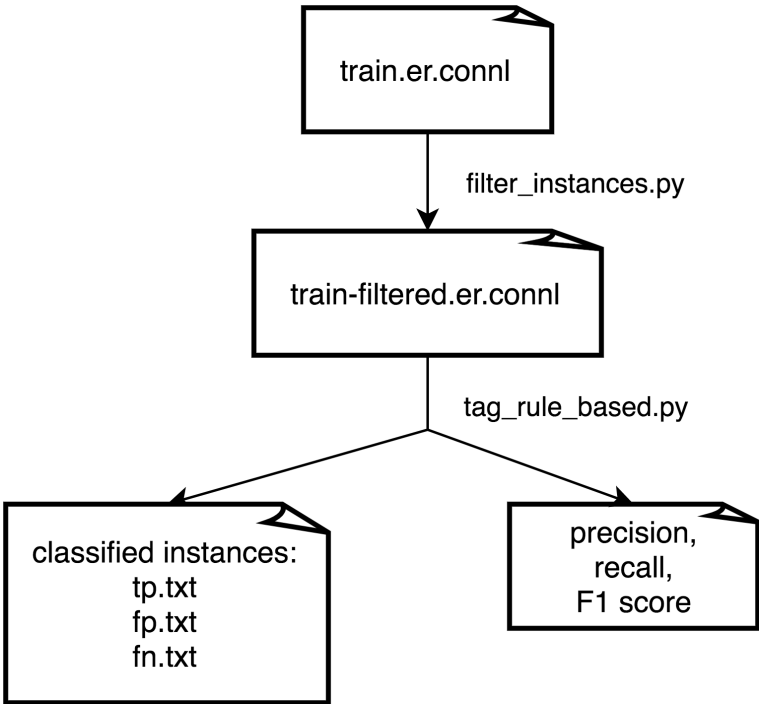

Figure 6.4: Evaluating rule based methods on police corpus

2. **PREDAT**: A word with lemma "předat" in any form is present between mentions of the entities.

3. **JEDNA_O**: Words "jedná" and "o" are present in this order between mentions of the entities.

4. **V_KONTAKTU**: Words "v" and "kontaktu" are present in this order between mentions of the entities.

In the following sections we describe the rules along with some anonymized examples as described in chapter 3.1.1.

Rule "A"

"A" means the conjunction *and* in Czech. If two persons are joined by the most common conjunction, they are clearly related, like in the following examples:

- NOVÁK a NOVOTNÝ za obchod s kokainem.
  NOVÁK and NOVOTNÝ for cocaine trade.

  This is a fragment of a text about who was convicted for what.

- Asi před týdnem mělo dojít k hádce mezi osobou NOVÁK a NOVOTNÝ ohledně OPL.
  About a week ago, there probably was a dispute between NOVÁK and NOVOTNÝ about drugs.

Rule "PREDAT"

"Předat" means *to hand over* in Czech. It is another significant trace of a relation - if somebody has handed over something (commonly drugs in the corpus) to somebody, they are likely to be related.

- Při prvním setkání s Novákem ve Sport baru v Plzni viděl, že Novák Jan předal cca 50 g OPL kokain Novotnému Josefovi.
  At the first meeting with Novak at the Sport Bar in Pilsen he saw that Novák Jan handed over 50 g of a drug cocaine to Novotný Josef.

- Dne 15.03.2002 měli NOVÁK a jeho spolupachatelka NOVÁKOVÁ předat 200 g drogy HEROIN ve městě BRATISLAVA/SLOVENSKO osobě NOVOTNÝ Jan, nar. 23.09.1982, přezdívka HONZA.
  On 15.03.2002 NOVÁK and his co-worker NOVÁKOVÁ hand over 200 g of HEROIN drug in BRATISLAVA/SLOVAKIA to a person NOVOTNÝ Jan, born 23.09.1982, nickname HONZA.

  We should note here that the rule detects only a relation between NOVÁKOVÁ and NOVOTNÝ, but not with NOVÁK, who is also related to both other persons.

Rule "JEDNA_O"

"Jednat se o" is a Czech verb phrase which can be freely translated as *to be* or *to be actually*. This Czech phrase is used to emphasize equivalence between two subjects over multiple sentences in a formal way, like *A person was arrested. He was a drug dealer.* – the *he was* can be formally expressed with *jednat se o* in Czech. In the police corpus the phrase is often used with parentheses instead of separate sentence, therefore we can use it to detect relations in a single sentence too.

"Jedná se o" is the present simple form of this phrase. Because the word order varies ("pravděpodobně se jedná o" = *it is probably* etc.), we check only for words "jedná" and "o" for simplicity. In the following examples you can see how this phrase is often used in the corpus.

- V případě osoby JOSEF, by se mohlo jednat o osobu známou pracovníkům agentury jako PEPA.
  In the case of JOSEF, he could be known to the staff of the agency as PEPA.

  "Mohlo by se jednat o" expresses a suspicion that JOSEF is the same person as PEPA, which is an important relation.

- Dále byl vytipován výrobce pervitinu pro osobu NOVÁK, jedná se o NOVOTNÝ Jan, nar. 15.01.1990/1234, bytem PRAHA 1, MALÁ STRANA, MALOSTRANSKÉ NÁMĚSTÍ 25.
  Furthermore, the producer of pervitin for NOVÁK was identified, it is NOVOTNÝ Jan, born. 15.01.1990 / 1234, apartment PRAGUE 1, SMALL SIDE, MALOSTRANSKÝ SQUARE 25.

  In this sentence we can see that "jedná se o" was found between NOVÁK

Rule	TP	TN	FP	FN	Precision	Recall	F1
A	193	15 068	136	3 342	0.587	0.055	0.100
V_KONTAKTU	51	15 170	34	3 484	0.600	0.014	0.028
PREDAT	17	15 194	10	3 518	0.630	0.005	0.010
JEDNA_O	7	15 188	16	3 528	0.304	0.002	0.004

Table 6.15: Results of rule based relation extractor

and NOVOTNÝ, which are not the same persons. However, the sentence actually expresses a relation between NOVÁK and NOVOTNÝ. We have found more sentences like this, but none of them contained a pair of persons that were not related, so we had no reason to consider this phenomenon to be a problem.

Rule "V_KONTAKTU"

"V kontaktu" means *in contact*. If two persons are in contact, they are obviously related.

- NOVÁK je v těsném kontaktu s osobou NOVOTNÝ Jan, nar.01.02.1991 /1234, trvale bytem OSTRAVA, OSTRAVSKÁ 12, MORAVSKOSLEZSKÝ KRAJ a lze předpokládat, že od osoby SMITH nakupují OPL společně.
  NOVÁK is in close contact with the person NOVOTNÝ Jan, born 01.02. 1991/1234, permanent apartment OSTRAVA, OSTRAVSKÁ 12, REGION MORAVSKOSLEZSKÝ and it is possible to assume that they buy drugs together from SMITH.

- AHMAD Adam je v kontaktu s osobou přezdívanou JOSEF, PEPA (ztotožněn jako JOSEFOVIČ Josef, nar. 01.12.1985, který užívá motorové vozidlo BMW, "Registrační značka" (*national registration number of a vehicle*) (RZ) ABC123 (ABC1234).
  AHMAD Adam is in contact with a person nicknamed JOSEF, PEPA (identified as JOSEFOVIČ Josef, born on December 1, 1985, who uses the BMW, RZ ABC123 (ABC1234).

6.7.2 Results of rules evaluation

All these four rules were 100% successful on the 500 sentences we have manually tagged. However, when we applied these rules on the corpus tagged only by the police, the precision was significantly lower as you can see in Table 6.15. This led us to a suspicion that the corpus contains too many sentences describing a relation where the relation is not tagged, so the evaluation of both machine learning and rule based methods gave such bad results.

7. Review of corpus quality

None of described experiments with machine learning or rule-based methods gave results good enough to be used in practice. Due to that, we decided to take a more detailed look at the corpus and consult the police officers who work with the corpus what they actually consider to be a relation between entities.

7.1 Possible problems of the corpus

Our assumption at this point was that there is too much noise in the relation tagging in the corpus.

One problem might be that there are many pairs of entities that are, more or less obviously, related in some sentences, but the relation was not interesting enough for the police or they just did not ever need to mark it.

Another problem could be the opposite – the same pair of entities can appear in multiple sentences and there is no information in the corpus telling in which sentence the relation is actually expressed. So, there can be sentences containing a related pair of entities, however not expressing the relation at all.

Both these phenomena could affect results of machine learning. The first one may also affect results of evaluating rule based methods: actual relations that are not tagged would falsely lower precision.

7.2 Results of review

The result of the detailed review is surprising. Not only the corpus tagging quality is seriously bad and many relations that are obviously expressed in the text are missing, but we identified that we can formulate several very simple rules based just on types of entities present in a sentence with precision close to 100%.

Measuring actual precision and recall of such rules is not possible automatically on the corpus because of the problem with missing tagging on relations, but we have examined over 200 random sentences from the corpus and did not identify any exception from these rules.

These rules are described in chapter 8.

7.3 What does a relation mean

Due to these problems with corpus quality, we have discussed with the police deeper what a relation between two entities means, or by what pattern they decide whether to tag a relation between a pair of entities or not.

The result of this discussion was that there are no explicit rules like “persons are related iff they are suspected to collaborate in a criminal activity”, “person is related to a car if he or she owns it” etc. They actually tag a relation where the connection between two entities may be *important*, which means *potentially useful* in watching criminal activities and convicting suspects.

For example if two people were in the same court process, it may or may not be relevant that they are related. If a car was seen at an address, it is not necessary to tag a relation if the address is a place of residence of the car's owner.

One may argue that any occurrence of two entities together in a document may be a *potentially useful* information. However, the practical sense of the relations tagging in the corpus is to quickly find *the most relevant* relations of an entity. Examining the corpus and consulting the police about what is and what is not a relation shown that many documents contain for example summary reports of some department's activity in a period of time. If two persons appear in different parts of such document, it only means that they were somehow connected with (not even the same) criminal activity at the same time. That information is not anyhow important for the police.

In theory, it may be helpful to use some kind of weighted relations, where simple presence of entities in a single document would give low-weight relations and entities present in defined patterns of significance would give high-weight relations. However, the corpus currently does not contain such information so we have no resources to experiment with such approach.

8. Rules identifying relations

In Section 6.7 we have described several rules that we were trying to develop to achieve best results on the tagged police corpus. However, based on the review of the corpus described in Chapter 7, we have found that the rules may be actually even simpler, just based on the presence of two entities in a sentence. Evaluating the rules using the tagged corpus is not relevant as tagging of the relations is significantly incomplete.

Therefore we have developed a set of simple rules that identify many more correct relations including those missing to be tagged in the corpus, consulting their correctness with policemen who use the corpus in practice.

Unlike the "traditional" relation extraction task like for example the one solved by kernel methods in Culotta and Sorensen [2004], which identifies particular semantical relations like "X is located at Y", "X is member of staff role Y", "X is citizen of Y", "X is founder of Y" etc., the police corpus defines the tagged relations only as a generic connection between the entities.

It is good enough for police purposes, because they use the corpus for examining which suspected or convicted persons are connected, which phone numbers they use / contact etc. Details about the connection, for example if one person is selling drugs to the other, can be easily found in the document in which the relation is tagged, and the corpus is not so large to filter relations of a person or other entity by type before examining relevant documents in detail.

These rules are based only on types and numbers of entities in a single sentence. The assumption that if two entities are present in the same sentence, they are most likely somehow related, was confirmed on a sample of over 200 sentences containing about 500 entity pairs. However, there are some situations in which we can say that the relation of entities cannot be determined from a sentence where these entities appear. These situations are described in the sections below.

The following sections contain a number of examples from the real police corpus anonymized as described in section 3.1.1.

8.1 Relations with Persons

8.1.1 Person – Person

When two or more persons appear in a sentence, they are always considered to be related. We have not found any exception from this rule in the corpus. One important reason is that persons are the most important entities in the corpus – they are the suspected or convicted people and connections between them are the most important for policemen. When two or more persons appear in a single sentence, it means that they figure in the same criminal case, which is a fact strong enough to consider them to be related.

Examples of such sentences:

- Jmenovaný se společně s osobami NOVÁK Jan a NOVOTNÝ Pavel podílel na ilegálním transportu 10 kg chlorhydrátu KOKAINU.
  The named person together with NOVÁK Jan a NOVOTNÝ Pavel con-

tributed to the illegal transport of 10kg of Cocaine Chlorohydrate.

An unspecified person is related to two confederates NOVÁK and NOVOTNÝ, which can be considered to be related.

- Státní příslušník POLSKO SMITH John a státní příslušník SPOJENÉ STÁTY AMERICKÉ JACKSON Jack byli zatčeni během odděleného vyšetřování prováděného Federální policií POLSKO a ABC, v jehož průběhu se pokusil NGUYEN CHI THI vyjednat zásilku 10 kg HEROINU tajnému agentu ABC.
  Polish citizen SMITH John and US citizen JACKSON Jack were arrested during a separate investigation conducted by the Federal Police of Poland and ABC, during which NGUYEN CHI THI attempted to negotiate a shipment of 10 kg of heroin to a secret agent ABC.

  The two persons were arrested there because of their connection to NGUYEN CHI THI, so all three persons mentioned in the sentence are related.

8.1.2 Person – Other entity

If there is exactly one person together with other types of entities, the person is related to these entities, but the entities are not related to each other. As described in Table 3.2, other possible entities are *Phone*, *Address*, *Car*, *Firm*, *Account*.

These entities appear in contexts like:

- a person uses or was contacted from a phone number,
- a person is living or was doing something on an address,
- a person owns or was going in a car,
- a person owns, work for, cooperates with a firm
- a person owns, receives money from or sends money to a bank account.

There were not found any sentences in the examined sample of the corpus that would break this rule. Examples of sentences conforming this rule:

- NOVÁK Jan dovezl ze státu ČESKÁ REPUBLIKA do státu NĚMECKO drogu MDMA v množství 20 g, kterou dne 04.11.2001 přechovával v okrese Aachen v osobním motorovém vozidle ŠKODA FELICIA, RZ ABC1234.
  NOVÁK Jan imported 20g of MDMA drug from CZECH REPUBLIC to GERMANY, which he kept in a passenger motor vehicle ŠKODA FELICIA, RZ ABC1234 in district Aachen on 4th November 2001.

  It is not clear if NOVÁK is the owner of the vehicle from the sentence, however, he certainly used the vehicle for his criminal activity, which is considered as a relation.

- Již při telef. kontaktu sdělil, že zásadní informace v tomto směru má jeho neteř NOVÁKOVÁ Jana, nar. 14.5.1990, tel. 420123456789, současně bytem na adrese KOMÁROVÁ 123, KOMÁROV.
  During phone contact, he said that essential information about this issue had his niece NOVÁKOVÁ Jana, born 14.5.1990, phone 420123456789, current address of residence KOMÁROVÁ 123, KOMÁROV.

  This example shows that Nováková Jana is related to a phone number (which she is using) and to an address (where she is living), but relation between the phone and the address would not make sense.

- Dalším šetřením v NON STOP baru THE BAR, BRNO, bylo zjištěno číslo MT na provozního NOVÁK Jan – 420123456789.
  During following investigation in NON STOP bar THE BAR, BRNO, phone number of the operating manager of the bar NOVÁK JAN was found out – 420123456789.

  This is another example where the person Novák is employed in the firm THE BAR and owns the phone number, so the person is related to both entities, but the phone number has no direct connection to the firm.

If there are multiple persons in a sentence, relations to other present entities cannot be automatically determined. For example if two persons are mentioned in a sentence along with a phone number, they may have contacted each other using that phone number, or just one person is using that number and the other person is not related to the number at all. Some examples of such sentences follow:

- Osoby byly ztotožněny jako NOVÁK Jan, nar. 10.01.1980/0123, NOVÁ LHOTA 10, NOVOTNÝ JOSEF, nar. 03.08.1990/0111, STARÁ LHOTA 123 a SKOČDOPOLE Petr, nar. 05.03.1980/1885, HORNÍ LHOTA 111.
  The people were identified as NOVÁK Jan, born 10.01.1980/0123, NOVÁ LHOTA 10, NOVOTNÝ JOSEF, born 03.08.1990/0111, STARÁ LHOTA 123 and SKOČDOPOLE Petr, born 05.03.1980/1885, HORNÍ LHOTA 111.

  This is an example of a sentence mentioning multiple persons and addresses. In this case it is easily understandable for a human reader which person belong to which address, however similar lists of people are represented in the corpus using many different syntaxes as shown in following examples, so it would need to properly annotate a big part of the corpus to determine and evaluate possible rules for extracting such relations.

- Vozidlo RZ 1A23456 je registrováno na osobu NOVÁKOVÁ Jana, nar. 25.12.1991/0123 Praha, bytem PRAHA 1, STARÉ MĚSTO, STAROMĚSTSKÉ NÁMĚSTÍ 1234/1, svobodná, sestra osoby NOVÁK Jan.
  Vehicle RZ 1A23456 is registered to person NOVÁKOVÁ Jana, born 25.12. 1991/0123 Praha, address of residence PRAHA 1, STARÉ MĚSTO, STAROMĚSTSKÉ NÁMĚSTÍ 1234/1, single, sister of person NOVÁK Jan.

  For a human reader there is obviously no direct connection between NOVÁK

and the vehicle, but there are not many similar sentences in the corpus to form a rule good enough to identify a significant number of relations.

- V minulosti byl do nedovoleného obchodu s heroinem zapojen NOVÁK Jan, nar. 10.10.1950, občan BULHARSKO, bydlel na adrese PRAHA 2, PRAŽSKÁ 123/45 u osoby NOVOTNÁ Jana, nar. 13.12.1985/1234, užívá telefon 420723012345.
  In past, NOVÁK Jan was involved in illicit trade in heroin, born 10.10.1950, citizen of BULGARIA, lived at address PRAHA 2, PRAŽSKÁ 123/45 with NOVOTNÁ Jana, born 13.12.1985/1234, using the telephone number 420723012345.

  This example shows that the relations are not so clear even for a human reader. When reading such a sentence, one can think that the phone number 420723012345 belongs to NOVOTNÁ Jana, because she appears more recently in the sentence and there is no syntactic clue if the number belongs to NOVOTNÁ or NOVÁK. We can estimate that it actually belongs to NOVÁK only from the fact that the document containing this sentence is probably describing all available information about NOVÁK. However, we can see that any simple relation extraction rules like "the person that is closer to the phone in the sentence is its owner" can be easily broken by such sentences, producing false positive results.

8.1.3 Person – Firm

Although it may seem that firms would make relations as properties of persons like for example cars and phone numbers, it shows that firms behave sometimes like persons in the corpus.

It is because the corpus describes cooperations of firms mostly in criminal activities related to drugs, so if they are mentioned together in a sentence, it means that the firms are connected, and if there are persons mentioned in such sentences, they represent the connection between the firms.

If there are multiple firms alone or with one or more persons in a single sentence, it means that they are all related together in some cases:

- Pro osobu NOVÁK ve městě PRAHA nadále distribuuje kokain osoba PEPA (blíže neztotožněna) a to v podnicích AHOJ, NAZDAR, HOSPODA, SLUNCE.
  For the person NOVÁK, in the city PRAHA, cocaine is also distributed by PEPA (not conditioned closer) in businesses AHOJ, NAZDAR, HOSPODA, SLUNCE.

  This sentence shows that PEPA is distributing drugs in several businesses (tagged as firms in the corpus), but NOVÁK is also connected to these companies - his drugs are distributed in these businesses, which is a connection strong enough for policemen to mark it as a relation in the corpus, even when they are distributed via intermediary PEPA. The businesses are also considered all to be related together in the drug affair described in the document.

- V lednu 2005 kolem 01:00 hodin přivezl NOVÁK Jan v doprovodu jmenované osoby NOVOTNÝ Josef do svého podniku AHOJ, BRNO, BRNĚNSKÁ 12/34, který pro NOVÁKA zde provozoval zmíněný HORAK Petr, nar. 10.01.1970 přezdívaný HORA, zásilku asi 5 kg HEROINU.
  In January 2005 at about 01:00 o'clock NOVÁK Jan accompanied by the named person NOVOTNÝ Josef brought in to his business HELLO, BRNO, BRNĚNSKÁ 12/34, operated here for NOVÁK by mentioned HORAK Petr, born 10.01.1970 nicknamed HORA, a shipment of approximately 5kg of HEROIN.

  Multiple people met in a business operated by another person to hand some drugs, so all mentioned people and the firm can be considered as related.

- NOVÁK Jan se v této době nachází na statku v obci KLADNO, kde pracuje pro osobu NOVOTNÝ Pavel, nar. 20.10.1980, který statek vlastní v rámci obchodní společnosti STATKY s.r.o.
  NOVÁK Jan is currently on a farm in the city KLADNO, where he works for NOVOTNÝ Pavel, born 20.10.1980, who owns the farm under a company FARMS s.r.o.

  One person owns the farm and the other works there for him, so both persons and the firm owning the farm are closely connected.

However, this assumption is rebutted by some other sentences describing other kinds of situations:

- NOVÁK Jan, alias "HONZA", nar. 10.01.1960/1234, bytem PÍSEK, U CEMENTÁRNY 100, okres PÍSEK, tel. 420123456789, který v minulosti jezdil jako závodník na lyžích a provozuje night klub ROSE, v minulosti také JANA a ROZITA v PARDUBICÍCH.
  NOVÁK Jan, alias "HONZA", born 10.01.1960/1234, address of residence PÍSEK, U CEMENTÁRNY 100, district PÍSEK, tel. 420123456789, who was a professional skier in past and is operating the night club ROSE, in past also JANA and ROZITA in PARDUBICE.

  NOVÁK is operating all three mentioned night clubs, however there is no other connection between them than NOVÁK itself, so they should be related in the corpus only indirectly through NOVÁK.

- K povolání NOVÁK Jan uvedl, že v roce 1999 byl údajně zaměstnán v autoservisu AUTOAUTO, PRAHA 10, SERVISNÍ, blíže neuvedeno a v roce 2000 AUTOBAZAR STARÁ AUTA, blíže neuvedeno.
  NOVÁK Jan said about his occupations that in 1999 he was supposedly employed in car service AUTOAUTO, PRAHA 10, SERVISNÍ, not specified any more details, and in year 2000 in AUTOBAZAR OLD CARS, not specified any more details. The fact that a (suspected) person was employed in some companies in the past certainly does not imply that the companies are anyhow connected.

- O firmě DUGONG s.r.o. a jejím řediteli, osobě NOVÁK Jan, nar.01.01. 1980, a o firmě ELEKTRONIKA PRAHA a jejím odpovědném vedoucím, osobě NOVOTNÝ Pavel se v našich systémech nevyskytují záznamy. *About the company DUGONG s.r.o. and its director, person NOVÁK Jan, born 01.01.1980, and about the company ELECTRONICS PRAGUE and his accountable manager, person NOVOTNÝ Pavel, there are no record in our systems.* There are mentioned two firms and a responsible person for each of them, but there is no connection between a responsible person with the other firm. We can only confirm that the *person – person* rule holds, because the mentioned persons with their firms are being investigated in the same criminal case.

If there are one or more firms with exactly one person in a sentence, the person is related to these firms, but we cannot say anything more about relations between the firms or other persons.

However, if there are multiple persons and exactly one firm in a sentence, the firm is related to all of these persons. We have not found any example rebutting this rule. It seems that when multiple persons are mentioned along with a single firm, it always means that they have some common activity related to the firm.

There were found no sentences like "Person X, owner of firm Firm Y, deals drugs to Person Z", which would contain multiple persons and only one of them related to the mentioned firm. Examples of sentences where multiple persons are related to one firm follow:

- NOVÁK PETR, nar. 1980 a NOVÁK Jan, nar. 1982, bytem PARDUBICE, PARDUBICKÁ ?, provozovatelé HERNA BAR "U PROHRANÉ VÝPLATY" v PARDUBICÍCH, PRAŽSKÁ ul.
  NOVÁK PETR, born 1980 and NOVÁK Jan, born 1982, address of residence PARDUBICE, PARDUBICKÁ ?, operators of CASINO BAR "LOST PAYOUT" in PARDUBICE, PRAŽSKÁ street

  This is a simple example of two people who are running a bar.

- V blíže nezjištěné době, kdy byl provozován BAR NA MIZINĚ osobami české národnosti, tj. od 2000 do cca 2002 bylo v bytě u blíže neztotožněné osoby, oslovované jako PEPA ve městě JIHLAVA, za přítomnosti osob NOVÁK Jan, NOVOTNÝ Pavel, nar. 20.01.1980 a jeho družky NOVÁ Pavla, nar. 15.03.1990 a SMITH John, nar. 10.01.1990, upravováno nezjištěné množství heroinu pro distribuci.
  At an unspecified time when the BANKRUPT BAR was operated by persons of the Czech nationality, from 2000 to 2002, in the apartment of an unidentified person called PEPA in the town of JIHLAVA, in the presence of persons NOVÁK Jan, NOVOTNÝ Pavel, 20.01.1980 and his partner NOVÁ Pavla, born 15.03.1990 and SMITH John, born 10.01.1990, there was processed an unknown amount of heroin for distribution.

  Although there was obviously not much information, we can see a connection between mentioned people and the bar, which was somehow related to their business.

- Z tohoto spisového materiálu bylo zjištěno, že Jan NOVÁK přijel v blíže nezjištěné době v měsíci březen 2002 společně s osobou NOVÁ Jana, nar. 23.15.1980 do hotelu MODRÁ HVĚZDA, kde společně kouřili marihuanu, kterou si s sebou přivezl NOVOTNÝ Pavel, když všechnu marihuanu vykouřili, odešel NOVOTNÝ Pavel z místnosti ze slovy, že se jde podívat na zásoby.
  From this record material it was found that Jan NOVÁK arrived at a later unspecified time in March 2002 together with the person NOVÁ Jana, born 23.15.1980 to the hotel BLUE STAR, where they smoked the marijuana that NOVOTNÝ Pavel brought with them, when they smoked all the marijuana, NOVOTNÝ Pavel left the room, saying that he was going to check supplies.

  In this example we can see two persons, that were going to use drugs in a hotel, so they are both related to the hotel.

8.2 Other entity types

Rules identifying relations between other entity types – *Phone*, *Address*, *Car*, *Firm*, *Account* are significantly more difficult to identify than *Person – Person* and *Person – Other entity*.

The main reason is that the *Person* is the most important entity for the police, because the main agenda related to the corpus is to watch criminal activities of suspect persons and convict them later. All entities of other types carry only additional information about these persons – they could be together in a car, on an address, use a place of some firms for their criminal activity, use phones for communication and bank accounts for money coming from selling drugs, so relations between persons themselves and their additional properties contain the most important information.

Relations between some other types of entities are not so important for the police. You can see in Table 3.3 that there is a significantly lower number of relations of such type *Car – Phone*, *Phone – Address*, *Car – Account*. We assume the reason is that these relations can be meaningful only in specific cases (tracking a phone number, using a bank account when buying a car). The Anti-drug Department does not monitor these cases much, or at least does not use relation tagging in the corpus for these cases.

We will take a look at the most common types of such relations and show on examples where we can estimate a relation between entities based on their appearance in a single sentence. We can look again to Table 3.3, where we can see what types of such relations are common in the corpus and which are not.

8.2.1 Phone – Phone

There are many examples of sentences containing an important relation between two phone numbers, mostly if communication between these phone numbers may be a clue of connection between even unknown criminals.

- Dne 01.01.2000 ve 12:40:35 h telefonoval ze svého tel. 420777123456 SMITH (nebo SMITTH) John, nar. 15.02.1990 na bulharské číslo 9991234567,

přičemž k hovoru nedošlo.
On January 1, 2000 at 12:40:35 pm SMITH (or SMITTH) John, born 15.02.1990 telephoned from his phone number 420777123456 to the Bulgarian number 9991234567, but the talk did not happen.

This is an example of an important relation between two cell phones. The attempted call to that number is obviously somehow important for the investigated activity of SMITH John and the relation between the numbers may actually be helpful for police investigation.

- Majitel telefonu IMEI 1234567890123, telefonní číslo 420777123456
Owner of phone IMEI 1234567890123, phone number 420777123456

  We can see here that police is also tagging International Mobile Equipment Identity (IMEI) (*International Mobile Equipment Identity*) numbers as an entity of type *Phone*. There is one simple rule we have found for these situations: if there are exactly two entities of type *Phone* and one of them is IMEI, it is a relation. The fact that it is IMEI is not explicitly tagged in the corpus, but it can be derived from the length of the textual representation of the number - if it is 15 digits long, it is an IMEI[1].

- Následně dne 01.01.2005 opět kontaktoval uživatel mobilní telefonní stanice č. 420777123456, NOVÁK Jan, uživatelku mobilní telefonní stanice č. 420777654321, NOVÁ Jana, kdy obsahem hovoru je diskuze o tom, kdo komu volal.
Subsequently, on January 1, 2005, the user of mobile telephone station No. 420777123456, NOVÁK Jan, contacted again user of mobile telephone station No. 420777654321, NOVÁ Jana, when the content of the conversation was a discussion of who called who.

  This sentence probably comes from an investigation of how suspects are connected through phone numbers, so these two phone numbers are tagged as related.

However, we can also find many examples of sentences with multiple phone numbers without a relation, mostly if there is a list of persons with their phone numbers (and no important connection directly between the numbers), or a list of phone numbers alone.

- NOVÁK nyní používá 420777123456, NOVOTNÝ 420777654321.
NOVÁK now uses 420777123456, NOVOTNÝ 420777654321.

  This is a simple example of a list of two persons, each of them owns a phone number, but there is no relation between these phone numbers.

[1] According to recommendation E.164 of the ITU Telecommunication Standardization Sector, a worldwide phone number can be up to 15 digits long, however it is usually shorter at least in European countries and we have not found any phone number 15 digits long in the corpus. IMEI is always consisting of 15 digits, therefore we can assume that a sequence of 15 digits tagged as *Phone* is an IMEI.

- Byla zjištěna nová, postupně aktualizovaná, telefonní čísla používaná osobou NOVÁK Jan – 420777123456, 420777234567, 420777345678, 420777456789 a IMEI 123456789012345.
  New, continuously updated, phone numbers used by person NOVÁK Jan were found – 420777123456, 420777234567, 420777345678, 420777456789 and IMEI 123456789012345.

  A suspect person is using four phone numbers, but the relation between these numbers is not important, they are related only through NOVÁK Jan. It is interesting that they have not tagged a relation between a phone number and the IMEI to tell which number was being used on that mobile phone. Unfortunately, it cannot even be determined from the sentence by a human reader if the suspect had been switching SIM cards in the phone with this IMEI, was using one of these numbers with this IMEI or there was no relation between them at all. This means that if there are multiple phone numbers or IMEIs in a sentence, we cannot automatically determine a relation between them.

- NOVÁK Jan, nar. 20.03.1990/1234, státní příslušnost ČESKÁ REPUBLIKA, trvale bytem NOVÁ VES, NOVOVESKÁ 12, přezdívka "JOHNNY", tel. 420777123456, 420777654321
  NOVÁK Jan, born 20.03.1990/1234, nationality CZECH REPUBLIC, permanent apartment NOVÁ VES, NOVOVESKÁ 12, nickname " JOHNNY ", phone 420777123456, 420777654321

  This is another example of a sentence containing just two phone numbers that NOVÁK uses, but the relation between these numbers is not important at all – they are related only through NOVÁK.

8.2.2 Address – Firm

We have found that if there is exactly one entity of type *Address*, exactly one entity of type *Firm* and no other entities, the address always is related to the firm, usually as a residence of the firm. No sentences rebutting this rule were found.

- Zasilatel: NGUYEN BUSINESS LTD, adresa ČÍNA, CHITHOU, CHING CHING, JONG ROAD, místnost č. 200, provincie TJANG
  Sender: NGUYEN BUSINESS LTD, adresa ČÍNA, CHITHOU, CHING CHING, JONG ROAD, místnost č. 200, provincie TJANG

  This is a simple fragment of text recognized by Treex as a sentence. It is actually a simple case where only a company and its address of residence is mentioned, which is an information useful enough to be a relation for the police.

- Lustrací na Internetu zjištěno, že se pravděpodobně jedná o skladiště THE LARGE STORE, adresa VELKÁ BRITÁNIE, LONDON STREET 1, LONDON.
  It was found by an Internet lustration that it is probably the warehouse

THE LARGE STORE, address GREAT BRITAIN, LONDON STREET 1, LONDON.

This is another example of a simple sentence mentioning a company and its address of residence.

- Dle evidence CIS je zaměstnaný od 01.01.2000 ve společnosti PAPÍRNA s.r.o., 123 45 PRAHA 2, PRAŽSKÁ 1.
  According to CIS[2] records, he has been employed since January 1, 2000 in PAPÍRNA s.r.o., 123 45 PRAGUE 2, PRAŽSKÁ 1.

  There is an important relationship between the firm PAPÍRNA s.r.o. and the address in PRAGUE. Even through we do not know if it is a place of residence of the firm or just a branch, a suspect person is working on this address for this firm, which can be an important clue for police.

However, if there is another entity in the sentence, even of another type than *Address* and *Firm*, we cannot be sure about the relation, like in following examples:

- Osoba NOVÁK opustila dne 1.2.2005 v dopoledních hodinách území ČR, kdy bylo zjištěno, že prostřednictvím společnosti AIRLINES TICKETS došlo k nákupu letenky do státu NĚMECKO, konkrétně do BERLÍN.
  Person NOVÁK left the territory of the Czech Republic on 1.2.2005 in the morning, it was found that a ticket was purchased via AIRLINES TICKETS to the state of GERMANY, namely BERLIN.

  In this sentence, *Person*, *Firm* (AIRLINES TICKETS) and *Address* (the city BERLIN is tagged here as an *Address*) are mentioned. The firm is widely known for selling tickets (name AIRLINES TICKETS is changed due to anonymization). There is definitely no explicit connection between the firm and the destination city BERLIN, so we cannot consider this to be a relation.

- Dne 20. 05. 2005 oznámil PEPA podezřelému NOVÁKOVI že u TESCO v PRAHA, STAROMĚSTSKÉ NÁMĚSTÍ 1, na něho někdo čeká.
  On May 20, 2005, PEPA notified the suspected NOVÁK that someone was waiting for him at TESCO in PRAGUE, STAROMĚŠSKÁ NÁMĚSTÍ 1.

  A branch office of a food store is mentioned with its address there. It is clearly not interesting as a relation for police investigation that the food store has a branch office on the address, the important information in this sentence is about the place of an ongoing meeting of a suspect with somebody.

8.2.3 Address – Car

There is one important situation where a car is related to an address: the police is tracking movements of the car. We can find many examples of such relation in

[2]register of foreigners in Czech Republic

the corpus:

- Dne 20.5.2002 v 12:34 hod. kontrolován ve vozidle FORD FOCUS, RZ: ABC1234, v k.ú. obce PŘELOUČ, okr. PARDUBICE.
  On May 20, 2002 at 12:34 checked in vehicle FORD FOCUS, RZ: ABC1234, in the area of PŘELOUČ, district. PARDUBICE

  This is a simple text fragment with an information where the vehicle was stopped and checked.

- v 15.12 hodin všichni společně přijeli vozidlem MAZDA 323: AB123CD na BČS SHELL – ROZVADOV, ul. ROZVADOVSKÁ 987.
  At 15:12 they all came by the vehicle MAZDA 323: AB123CD to gas station SHELL – ROZVADOV, street ROZVADOVSKÁ 987.

  This is another example of an important place where the watched car had stopped.

- Dne 02.05.2002 ve 20:35 hod. bylo kontrolováno policejními orgány v SRN, 92714 PLEYSTEIN, na dálnici km 345.678, vozidlo FORD MONDEO 1.8 D, RZ 1A23456, VIN W123456ABCDEF123456, ve kterém cestovali:
  On May 2, 2002 at 20:35 it was checked by police institution in SRN, 92714 PLEYSTEIN, on the highway km 345.678, vehicle FORD MONDEO 1.8 D, RZ 1A23456, VIN W123456ABCDEF123456, in which these persons were traveling:

  In this fragment of sentence (cut by Treex on the terminating colon as a sentence) we can see another example of a watched movement of a car.

Fortunately, if the only entities in sentence are the *Car* and the *Address*, they are always related in the corpus – we have found no example rebutting the rule. However, if there are any more entities present, we cannot determine any relations. Most usually if there is a *Person* mentioned, it is unclear if he/she just owns the *Car* and appears on the *Address*, so the relation between the *Car* and *Address* is not relevant.

- Prověrkou v Centrálním registru vozidel bylo zjištěno, že RZ 1A23456 byly přiděleny vozidlu AUDI A6, barva stříbrná, provozovatel NOVÁK Jan, nar. 21.02.1982, bytem 123 45 ZADNÍ LHOTA 215, okr. BEROUN.
  By checking the Central vehicle registry it was found that RZ 1A23456 were assigned to AUDI A6, silver color, operator NOVÁK Jan, born 21.02.1982, address of residence 123 45 ZADNÍ LHOTA 215, district BEROUN.

  The relation between the vehicle and its operator's address of residence is not significant for the police.

8.2.4 Car – Firm

There are many similar occurrences of a *Car* and a *Firm* together, where the owner of *Car* is the *Firm*, like in the relation *Person – Car*:

- Cestovala z města BERLÍN / NĚMECKO v taxíku společnosti TAXITAXI, RZ ABC1234.
  She traveled from the city BERLIN / GERMANY in a taxi of company TAXITAXI, RZ ABC1234

However, there are also many sentences where the relation between a *Car* and a *Firm* is not relevant even through there are no other entities in these sentences:

- 11.20 hodin, nasedli do vozidla FORD FIESTA, RZ: 1A23456 a odjeli na parkoviště u prodejny TESCO.
  11:20 o'clock they mounted in a vehicle FORD FIESTA, RZ: 1A23456 and went to the parking lot at TESCO store.

  This is an example of a common issue of the relation extraction problem of this corpus: TESCO is a widely known company, and the car that stopped near its store is probably not related to the company in the manner of selling drugs. It is even disputable why it is important for the police to tag companies like TESCO in the corpus (actually it was another, similarly well-known company in the original text, changed during anonymization), however some police annotators still do it.

- Později toho dne byly sudy přivezeny modrou pronajatou dodávkou s RZ: 12ABCD, řidič dodávky zaplatil 500 eur za přepravu a předal list papíru s informacemi o adresátovi - JOHN DOE, tel. číslo 441234567890 (ANGLIE), DDD - společnost v LIVERPOOLU.
  Later that day, the barrels were brought with a blue leased van with an RZ: 12ABCD, the delivery driver paid 500 Euro for the shipment and handed over a sheet of paper with information about the addressee - JOHN DOE, phone number 441234567890 (ENGLAND), DDD - LIVERPOOL

  A van had delivered drugs probably to some mediator, which should have deliver it further to the DDD company. However, the connection between the van and the DDD company is too spread to be a relevant relation for the police.

8.2.5 Firm – Phone

While examining *Firm – Phone* relations we have found that police also tag web addresses as *Phone*. The reason is simple – the web address is also a form of (electronical) contact, and because of limitations of a number of entity types in other police software, they re-use the *Phone* entity also for web addresses.

- Webová adresa <www.example.com> patří rakouské společnosti EXAMPLE se sídlem RAKOUSKO, DORFEN, 123 Hauptstrasse.
  The web address <www.example.com> is owned by Austrian company EXAMPLE with residence AUSTRIA, DORFEN, 123 Hauptstrasse.

  Here we can see a simple binding between a web address as a *Phone* entity and a company.

- 43666123456 - firma SMITHSSON GmbH 1234 VÍDEŇ, Hauptstrasse 10
  43666123456 - company SMITHSSON GmbH 1234 WIEN, Hauptstrasse 10

  There are many fragments of text like this detected as a sentence in the corpus, containing only a firm and a phone number, which are obviously related.

It seems that if *Phone* and *Firm* are the only two entities in a sentence, the rule that they are related holds. However, there is a lot of sentences like the following, where a person is mentioned in the sentence along with the phone and firm, the *Phone* is not related to the *Firm*, however the person is not tagged as an entity.

So, we can apply such rule only under a circumstance that all entities are tagged in the sentence – in this case the noise of not tagged persons is very significant in the corpus.

- 420777123456 (uživatelka NOVÁ JANA - bývalá jednatelka fy THECOMPANY)
  420777123456 (user NOVÁ JANA - former executive of comp. THECOMPANY)

  Here we can see a simple sentence containing a *Phone*, *Firm* and *Person*, where the phone is in no manner related to the company. However, person was not even tagged in the sentence as an entity, so this is an example of the "noise" coming from sentences with missing tagging. Such sentence should not rebut a possible rule that a *Firm* with a *Phone* without other entities in a single sentence form a relation.

- Po příjezdu do města PRAHA zavolal NOVÁK neznámé osobě na telefonní číslo 777123456 a domluvil se, že si drogu převezme na benzínové stanici OMV na adrese ČESKÁ REPUBLIKA, PRAHA, PRAŽSKÁ 1.
  After arriving to the city of PRAGUE, NOVÁK called an unknown person at a phone number 777123456 and they agreed that he will take over the drug at gas station OMV at the address CZECH REPUBLIC, PRAGUE, PRAŽSKÁ 1.

  In this sentence, an address and a person are present along with *Phone* 777123456 and *Firm* OMV (well-known gas station). However, the address and the person is not tagged as an entity in that sentence.

8.2.6 Firm – Firm

Unfortunatelly we are not able to expect any relation between two firms. Even if two firms are the only entities in a sentence, they are not related in many sentences like these:

- PIZZERIE v místě bývalé restaurace SOKOLOVNA, BRNO - VENKOV /poblíž nádraží/.
  PIZZERIA at the place of former restaurant SOKOLOVNA, BRNO - CO-

UNTRYSIDE /near railway station/

The mention of restaurant SOKOLOVNA is there only to specify that PIZZERIA is where SOKOLOVNA used to be, with no clue that the firms are anyhow related.

- V souvislosti s touto garáží bylo dále zjištěno, že na tuto byla z hotelu MODRÁ HVĚZDA, který je umístěn naproti objektu restaurace VÝVAŘOVNA směrována kamera.
  In connection with this garage it was further found out that a camera was routed onto it from the MODRÁ HVĚZDA Hotel, which is located opposite to the object of the restaurant "VÝVAŘOVNA". This is another example where a firm (VÝVAŘOVNA restaurant) is used to specify a location of something, without any relation between the mentioned firms.

8.2.7 Car — Car

It seems that the only connection between two cars are mostly the persons using these cars and they should not be tagged as a direct relation. Although there are sentences describing relations between cars, i.e. "Jednalo se o tahač MERCEDES, zelená, RZ A123456 a návěs RZ A123457 s ruským řidičem jménem Vasil" (*It was a MERCEDES tractor, green, RZ A123456 and a semi-trailer RZ A123457 with a Russian driver named Vasil*), they cannot be simply distinguished from cases that are not relations based only on occurrence of entities in a sentence. These are examples of sentences where only two cars appear as entities, but they are not related:

- Opakovaně kontrolován ve vozidle FORD FIESTA červená barva RZ 2A34567 a FIAT MULTIPlA zelená barva RZ 3A45678.
  Repeatedly checked in a vehicle FORD FIESTA of red color, RZ 2A34567 a FIAT MULTIPlA zelená barva RZ 3A45678

  This sentence describes two vehicles belonging to a person not mentioned in the sentence. There is no direct relation between these vehicles other that their common owner.

- Tito muži byli následně pozorováni v modré dodávce s RZ: ABC2345 a/nebo ABC1234.
  These men were later spotted in a blue van with RZ: ABC2345 and/or ABC1234.

  The only connection between the two vehicles there are also the persons they were using them.

Other types of relations for which we can find both entities in a single sentence are represented in the corpus in too few instances (less than 100) to make any conclusions about automatically identifying such relations.

8.3 Excluded documents

There are many documents containing lists of suspects, phone numbers etc. These lists are often recognized as a single sentence by Treex (which is technically correct, because sentences can contain visually augmented lists of entities and still be considered to be a sentence). However, entities contained in such lists may or may not be related – we cannot be sure they are, even if they match the rules described in this chapter.

Such documents should be excluded from relation extraction using these rules. They can be recognized by several clues:

- These sentences contain a visual representation of list in plain-text, like row prefixes "a), b), c)", "1., 2., 3." etc.
- Such documents contains a keyword International Criminal Police Organization (INTERPOL) – reports from or to INTERPOL are often containing a large number of entities and the relations can be hardly determined from the text even for a human reader.

Anonymized example of a portion of such document containing a list follows.

```
INTERPOL Praha
Č.j.:  CZ-123456/7890                Praha 01.01.2000

ST-ZPRÁVA INTERPOLU  BERLIN
Č.j.: nové (přiděleno č.j. ABC123/2000)

1. únor - srpen  2000
2. přihraniční oblast DORF / NĚMECKO (byt) poblíž města AŠ,
    přihraniční oblast NĚMECKO - RAKOUSKO v okolí ÖTZTAL (byt)
3. 0 kg 24,1 g MARIHUANA, 152,21 g KOKAIN
4. a) NOVÁK Jan, 01.01.1990 PRAHA, muž, státní příslušnost
  ČESKÁ REPUBLIKA, bytem ČESKÁ REPUBLIKA, 110 00 PRAHA,
  HLAVNÍ 18,
b) NOVOTNÝ Josef, 04.01.1980 PRAHA, muž, státní příslušnost
  ČESKÁ REPUBLIKA, pobyt neznámý
c) NOVÝ Pavel, 20.10.1980, muž, státní příslušnost ČESKÁ
  REPUBLIKA, pobyt neznámý
d) NOVÁKOVÁ Judita, 20.01.1985 VARNA / BULHARSKO, žena, státní
  příslušnost BULHARSKO, posledně bytem BULHARSKO, 12345 VARNA,
  VARNSKA 8, současný pobyt neznámý

// english translation follows:

INTERPOL Prague
Reference number:  CZ-123456/7890              Prague 01.01.2000

ST-MESSAGE FROM INTERPOL BERLIN
Reference number: new (assigned r.n. ABC123/2000)

1. february - august  2000
2. border area DORF / GERMANY (appartment) near AŠ city,
    border area GERMANY - AUSTRIA near ÖTZTAL (appartment)
3. 0 kg 24,1 g MARIJUANA, 152,21 g COCAINE
4. a) NOVÁK Jan, 01.01.1990 PRAGUE, male, Nationality
  CZECH REPUBLIC, address of residence ČESKÁ REPUBLIKA,
```

```
  110 00 PRAHA, HLAVNÍ 18,
b) NOVOTNÝ Josef, 04.01.1980 PRAHA, male, Nationality
  CZECH REPUBLIC, address of residence unknown
c) NOVÝ Pavel, 20.10.1980, male, Nationality CZECH REPUBLIC,
  address of residence unknown
d) NOVÁKOVÁ Judita, 20.01.1985 VARNA / BULGARIA, female,
  nationality BULGARIA, last address of residence BULGARIA,
  12345 VARNA, VARNSKA 8, current address of residence unknown
```

Code Snippet 8.1: Message from the Interpol

You can see from the document that it is a summary of important events, persons, found drugs etc. from a time range reported between police departments of two countries, however we can hardly recognize any information from such documents without knowing a deeper context.

8.4 Overview of applicable rules

Table 8.1 briefly recapitulates all rules described in this chapter. These rules may be used to identify relations in the corpus with confidence close to 100% – although it is not proven that they always hold, we have not found any sentences rebutting these rules in the available corpus.

Entity pair	Conditions of relation
Person – Person	All pairs of *Person*s found in the same sentence are related.
Person – Firm	If there is exactly one *Firm* in a sentence with one or more *Person*s, all these *Person*s are related to the *Firm*.
Person – Other	If there is exactly one *Person* along with other types of entities, the *Person* is related to all these entities.
Phone – Phone	If there are exactly two entities of type *Phone* and one of them is IMEI (15 digits number), they are related. Entities of other types can be present in the sentence, but more *Phone*s can not.
Firm – Address	If there is exactly one *Firm* and one *Address* and no other entities, the *Firm* and *Address* are related.
Car – Address	If there is exactly one *Car* and one *Address* and no other entities, the *Car* and *Address* are related.
Firm – Phone	If there is exactly one *Firm* and one *Phone* and no other entities, these are related *assuming that all mentions of people and other potential entities are tagged in the sentence, which is not true for many sentences in the corpus.*

Table 8.1: Rules identifying relation based on occurrence in the same sentence

In other cases, we cannot decide if entities not matched by these rules are related or not – we have found no "negative rules" determining circumstances under which entities are not related. We can only estimate that there will be no relations of entity types that were not found in the corpus regarding to table 3.3: *Car – Account*.

There is also another problem – there are many mentions of persons, cars etc. that are not tagged as entities at all. Most of the rules above rely on the fact that some types of entities are not present in the sentence to determine a relation. It means that it is important to have all entities tagged in a text to which these rules would be applied to extract relations.

Also, note that there are also well-known companies and franchises like TESCO (food stores), OMV (gas stations) etc. which are also tagged as an entity of type *Firm*. However, they are mostly mentioned only as meeting places of suspects (like "two suspects met at TESCO in PRAGUE") and relations to such firms are mostly irrelevant.

We suggest to filter out these well-known companies from any relation extraction, because even if they would have figured in sentences matching any rules mentioned in this chapter, there would probably be no important direct relation to them regarding criminal drug activity.

Conclusion

Exploration and experiments with the police corpus described in this work led us to the conclusion that neither machine learning methods, nor rule-based Natural language processing (NLP) are able to identify patterns and extract positive relations between entities using the set of entities and relations already tagged in the corpus as a source of rules or machine learning training data. We assume that experiments with machine learning and NLP rules gave results insufficient for practical usage from two main reasons.

First, not everything that should be tagged in the corpus is actually tagged. After reviewing several documents with people from the Anti-drug Department we have found that many pairs of entities that should obviously be related are not tagged as a relation, also there is a big number of mentions of people, addresses, firms etc. which are not even tagged as an entity. Such sentences are "confusing" machine learning methods and also distorting precision results of evaluation.

Second, there are no straightforward logical constraints defining what does it mean entities are related, like "Person is related to a firm if he/she works for the firm". The relation actually means that the entities are "somehow significantly related for police purposes" and the raw text of these documents how they are actually related. This means that it is difficult to identify grammar or syntactic patterns representing a relation. Even there may be multiple sentences with similar syntax, where one describes a relation and others do not according to the current data, and it cannot be decided without knowing the actual context of the sentence or document.

However, there can be defined several simple rules able to recognize a relation based on number and types of entities present in one sentence. These rules are summarized in Table 8.1. We have found no sentences rebutting these rules by examining random samples of the corpus, so we can estimate they are highly reliable, however because of errors and insufficiencies in the tagging of the available corpus we cannot say they are 100% sure.

Another difficulty was caused by the fact that when we started this work, we were told by police that they are starting to use *TextAn* for tagging entities and the only work we should cover would be developing a relation extraction system that could be used by *TextAn* as an external application. During our investigation we were told by the police later that they did not start to use *TextAn* on regular basis, because its entity recognition had to be trained on much more training data to be practically usable and they did not have capacities to provide such data.

Based on this acquired knowledge, we think that there are two options for police to (partially) automate relation extraction on their texts.

First, if they would be able to invest time into tagging a significant amount of documents, more experiments with machine learning or NLP could be done on the data. However, as they said, they have currently no human resources available for such work.

The other option is that they can implement several simple rules like these described in Table 8.1, try using them to suggest relations when tagging new documents, evaluate if it is helpful or not, and empirically improve these rules

over time.

As we know now that the police is negotiating about a contract for a new application for working with the corpus, and the rules we suggested in this work are technically simple, we have not developed an application that would apply these rules. Instead, we have summarized the rules we have found and suggested a simple Application Programming Interface (API) described in section 4.1 for an easy separation of the relation extraction problem out of such application into a stand-alone solution.

Bibliography

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries - DL '00*. Association for Computing Machinery (ACM), 2000. doi: 10.1145/336597.336644. URL http://dx.doi.org/10.1145/336597.336644.

Nguyen Bach and Sameer Badaskar. A review of relation extraction. Language Technologies Institute, School of Computer Science, Canergie Mellon University, Pittsburgh, PA 15213, November 2007.

Sergey Brin. Extracting patterns and relations from the world wide web. Technical Report 1999-65, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/421/. Previous number = SIDL-WP-1999-0119.

Hong-woo Chun, Yoshimasa Tsuruoka, Jin-dong Kim, Rie Shiba, Naoki Nagata, and Teruyoshi Hishiki. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. In *Proc. PSB 2006*, pages 4–15, 2006.

CoNNL-X. Conll-x shared task: Multi-lingual dependency parsing, 2006. URL http://ilk.uvt.nl/conll/#dataformat.

Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 423–429, Barcelona, Spain, July 2004. doi: 10.3115/1218955.1219009. URL http://www.aclweb.org/anthology/P04-1054.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll. In *Proceedings of the 13th conference on World Wide Web - WWW '04*. Association for Computing Machinery (ACM), 2004. doi: 10.1145/988672.988687. URL http://dx.doi.org/10.1145/988672.988687.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68, dec 2008. doi: 10.1145/1409360.1409378. URL http://dx.doi.org/10.1145/1409360.1409378.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex - relation extraction using dependency parse trees. Institut für Informatik, Ludwig-Maximilians-Universität München, Amalienstr. 17, 80333 München, Germany, December 2006.

Jan Hajič. Positional tags: Quick reference, 2000. URL http://ufal.mff.cuni.cz/pdt/Morphology_and_Tagging/Doc/hmptagqr.html.

Minlie Huang, Xiaoyan Zhu, Yu Hao, Donald G. Payan, Kunbin Qu, and Ming Li. Discovering patterns to extract protein — protein interactions from full

texts. *Bioinformatics*, 20(18):3604, 2004. doi: 10.1093/bioinformatics/bth451. URL `http://dx.doi.org/10.1093/bioinformatics/bth451`.

Linguistic Data Consortium. *Datasets for generic relations extraction (reACE, catalog number LDC2011T08)*. Linguistic Data Consortium, Philadelphia, Pa, 2011. ISBN 1-58563-582-0.

Dave Orr. 50,000 lessons on how to read: a relation extraction corpus. Product Manager, Google Research, April 2013.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. A review of relation extraction. *Journal of Machine Learning Research 3 (2003) 1083-1106*, February 2003.

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

BOM Byte-order mark

DIPRE Dual Iterative Pattern Relation Expansion

IMEI International Mobile Equipment Identity

INTERPOL International Criminal Police Organization

kNN k-nearest neighbor

NLP Natural language processing

NP Noun Phrase

NPList List of Noun Phrases

POS Part of speech

REST API Representational State Transfer Application Programming Interface

RZ "Registrační značka" (*national registration number of a vehicle*)

SVM Support Vector Machine

ÚFAL Institute of Formal and Applied Linguistics (Ústav formální a aplikované lingvistiky)

URL Uniform Resource Locator

List of Code Snippets

Attachments

1. Evaluation results on LDC corpus with simple feature set
2. Evaluation results on the police corpus with simple feature set
3. Definition and example of `.er.connl` format
4. Compact disc with scripts used for machine learning and rule based experiments

Attachment 1: basic feature set, LDC corpus

algorithm	total instances	positive instances	precision	recall	f1
svm	1000	126	0.66	0.57	0.61
svm	2000	181	0.45	0.42	0.42
svm	5000	344	0.38	0.30	0.32
svm	10000	681	0.52	0.33	0.40
svm	20000	1412	0.53	0.30	0.38
knn (1 neighbor)	1000	126	0.52	0.56	0.53
knn (1 neighbor)	2000	181	0.36	0.47	0.40
knn (1 neighbor)	5000	344	0.33	0.42	0.36
knn (1 neighbor)	10000	681	0.37	0.41	0.39
knn (1 neighbor)	20000	1412	0.42	0.42	0.42
knn (3 neighbors)	1000	126	0.57	0.56	0.57
knn (3 neighbors)	2000	181	0.48	0.46	0.45
knn (3 neighbors)	5000	344	0.46	0.40	0.41
knn (3 neighbors)	10000	681	0.53	0.36	0.42
knn (3 neighbors)	20000	1412	0.51	0.37	0.43
knn (5 neighbors)	1000	126	0.59	0.55	0.57
knn (5 neighbors)	2000	181	0.63	0.47	0.51
knn (5 neighbors)	5000	344	0.55	0.38	0.43
knn (5 neighbors)	10000	681	0.62	0.35	0.44
knn (5 neighbors)	20000	1412	0.58	0.34	0.43
knn (10 neighbors)	1000	126	0.68	0.52	0.59
knn (10 neighbors)	2000	181	0.71	0.42	0.49
knn (10 neighbors)	5000	344	0.65	0.35	0.44
knn (10 neighbors)	10000	681	0.72	0.30	0.42
knn (10 neighbors)	20000	1412	0.65	0.29	0.40
knn (20 neighbors)	1000	126	0.71	0.51	0.59
knn (20 neighbors)	2000	181	0.75	0.35	0.43
knn (20 neighbors)	5000	344	0.80	0.25	0.38
knn (20 neighbors)	10000	681	0.77	0.25	0.36
knn (20 neighbors)	20000	1412	0.70	0.23	0.35
knn (30 neighbors)	1000	126	0.81	0.43	0.56
knn (30 neighbors)	2000	181	0.91	0.29	0.40
knn (30 neighbors)	5000	344	0.81	0.15	0.26
knn (30 neighbors)	10000	681	0.80	0.22	0.34
knn (30 neighbors)	20000	1412	0.73	0.19	0.30

Attachment 2: basic feature set, police corpus

algorithm	total instances	positive instances	precision	recall	f1
svm	1000	89	0.20	0.15	0.15
svm	2000	116	0.12	0.08	0.07
svm	5000	238	0.16	0.16	0.15
svm	10000	252	0.24	0.12	0.08
svm	20000	316	0.23	0.11	0.06
knn (1 neighbor)	1000	89	0.09	0.11	0.09
knn (1 neighbor)	2000	116	0.10	0.13	0.10
knn (1 neighbor)	5000	238	0.18	0.23	0.19
knn (1 neighbor)	10000	252	0.29	0.19	0.15
knn (1 neighbor)	20000	316	0.24	0.15	0.08
knn (3 neighbors)	1000	89	0.16	0.09	0.10
knn (3 neighbors)	2000	116	0.17	0.09	0.10
knn (3 neighbors)	5000	238	0.23	0.17	0.18
knn (3 neighbors)	10000	252	0.31	0.13	0.11
knn (3 neighbors)	20000	316	0.27	0.11	0.08
knn (5 neighbors)	1000	89	0.22	0.09	0.11
knn (5 neighbors)	2000	116	0.13	0.04	0.06
knn (5 neighbors)	5000	238	0.19	0.09	0.12
knn (5 neighbors)	10000	252	0.04	0.07	0.04
knn (5 neighbors)	20000	316	0.26	0.07	0.04
knn (10 neighbors)	1000	89	0.07	0.03	0.05
knn (10 neighbors)	2000	116	0.24	0.03	0.05
knn (10 neighbors)	5000	238	0.17	0.05	0.07
knn (10 neighbors)	10000	252	0.27	0.06	0.06
knn (10 neighbors)	20000	316	0.28	0.03	0.03
knn (20 neighbors)	1000	89	0.00	0.00	0.00
knn (20 neighbors)	2000	116	0.01	0.01	0.01
knn (20 neighbors)	5000	238	0.08	0.01	0.01
knn (20 neighbors)	10000	252	0.01	0.01	0.01
knn (20 neighbors)	20000	316	0.01	0.01	0.01
knn (30 neighbors)	1000	89	0.00	0.00	0.00
knn (30 neighbors)	2000	116	0.01	0.01	0.01
knn (30 neighbors)	5000	238	0.00	0.00	0.00
knn (30 neighbors)	10000	252	0.00	0.00	0.00
knn (30 neighbors)	20000	316	0.01	0.01	0.01

Attachment 3: .er.connl format

Field number	Field name	Description
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma of word form, or an underscore if not available.
4	CPOSTAG	Coarse–grained part–of–speech tag. *unused by our application, as Czech POS tag (next field) contain both coarse–grained and fine–grained POS type.*
5	POSTAG	Czech positional part–of–speech tag as described by Hajič [2000].
6	FEATS	Unordered set of syntactic and/or morphological features. *unused by our application.*
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language.
9	ETYPE	Entity type, like *person*, *car*, *address* etc.
10	EID	Entity id from corpus; any string, serves to identify mentions referring to the same entity–relation
11	RELATIONS	comma separated list of token IDs (number first column) representing entities related to this one

Example sentence

```
@ABC1532 s5
1 here  here  -  RB  2 s
2 is    be    -  VBZ 0
3 abc   abc   -  NNP 6 gen ORG ABC1532_62 5
4 's    's    -  POS 3 poss
5 aaron aaron -  NN  6 nn  PER ABC1532_61
6 brown brown -  NN  2 pred
7 .     .     .  .   0
```