ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)

στα ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

“Εξαγωγή Σχέσεων από Κείμενα του Παγκόσμιου Ιστού”

Τσιμπίδη Δωροθέα-Κωνσταντίνα

M315009

ΑΘΗΝΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2017

MASTER THESIS

"Text-based Relation Extraction from the Web"

Tsimpidi Dorothea-Konstantina

SUPERVISORS: **Anastasia Krithara,** Research Associate, NCSR "Demokritos"
Ion Androutsopoulos, Associate Professor, AUEB

February 2017

ACKNOWLEDGMENTS

I would first like to thank my thesis advisor Anastasia Krithara for the constant guidance that she offered me and for allowing me to grow as a research scientist in the area of Information Extraction.

I would also like to thank the AUEB Natural Language Processing Group and especially professor Ion Androutsopoulos for the valuable feedback every time it was needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

There exists a gigantic amount of unstructured information on the Web, including blogs, email systems, social networks and so on. As a result, the need to produce structured information in order to interpret those data is rising. Although the need of interpretation is undeniable, it is impossible for humans to annotate all the semantic information that lies in that text. Information extraction (IE), which has received increasing attention in recent years, is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. Structured knowledge is required in order to effectively manage, summarize, search and find relevant information in large amounts of heterogeneous data. In most of the cases, this activity concerns processing human language texts by means of natural language processing (NLP). In information extraction entities such as people, organizations, locations or even genes and proteins can be extracted. Relations between those entities are extracted as well. That task, which is a sub-field of information extraction, is called relation extraction (RE).

Many different methods and approaches to relation extraction have been proposed over the years. However, those approaches face challenges regarding the definition of what is considered a relation and the variety of relation forms.

In this thesis we propose a new approach called CombIE that attempts to solve the aforementioned problem of converting unstructured Web text to structured. CombIE differs from previous systems in the sense that it combines two different approaches, rules and machine learning, in order to create a system that operates in two different ways.

The outline of the thesis is as follows: Chapter 2 analyzes the state of the art. Chapter 3 describes the system that was built in the scope of this thesis. In Chapter 4, the evaluation datasets, the metrics and the experiments are reported. Section 5 concludes with a summary and discussion of future work.

Chapter 2

State of the Art

Relation extraction is the task of finding semantic relationships between a set of entities in text. Relation extraction approaches can be divided into two categories: Traditional Relation Extraction and Open Relation Extraction approaches. In Traditional RE, the relation of interest has to be specified in advance while in Open RE, various relations can be extracted without requiring any prior knowledge.

Each relation consists of two or more arguments and a predicate. Most relation extraction systems focus on extracting binary relations, which consist of one predicate and two arguments, a subject and an object. Relations that are not binary are called n-ary and consist of one predicate and n entities, which usually contain much more information than the binary ones.

An example of binary relation is the *located-in(AUEB, Athens),* where "located-in" is the predicate, "AUEB" is the subject and "Athens" is the object. Another example is the sentence "At codons 12, the occurrence of point mutations from G to T were observed", in which there exists a 4-ary biomedical relation. The biomedical relationship between a type of variation, its location and the corresponding state change from an initial-state to another and can be expressed by the relation *point-mutation(codon, 12, G, T),* where "point-mutation" is the relation's predicate.

2.1 Traditional Relation Extraction

In Traditional RE the target relation has to be specified in advance. That relation is given to the RE system as input along with hand-crafted extraction patterns or patterns learned from hand-labeled training examples. When the need arises to shift to a new relation, humans are required to manually create new extraction patterns or specify new training examples. This manual labor scales linearly with the number of target relations.

Although RE has become increasingly automated over time, enumerating all potential relations of interest for extraction by a system is highly problematic for corpora as large and varied as the Web. To make it possible for users to issue diverse queries over heterogeneous corpora, RE systems had to evolve from architectures that required relations to be specified in advance.

2.2 Open Relation Extraction

Open RE systems address the problem mentioned above by extracting relations from corpora without requiring domain-specific knowledge and target relations in advance. Open RE's extraction process is linear in the number of documents in the corpus and is ideally suited to corpora such as the Web, where their relations are not known in advance and their number is massive.

A large number of Open Relation Extraction approaches have been proposed, including machine learning algorithms [1], rules and heuristics [2], shallow parsing [3], dependency parsing [4] and semantic role labeling (SRL) [6].

Shallow parsing methods annotate the sentence with part-of-speech (POS) tags, where POS tagging is the process of marking up a word in a corpus as corresponding to a particular part of speech, such as nouns, verbs, adjectives, adverbs, etc.

Shallow parsing methods analyze sentences by first identifying their constituent parts (part-of-speech (POS) tags, e.g. nouns, verbs, adjectives, etc.) and then links them to higher order units that have discrete grammatical meanings (noun phrases, verb phrases, etc.).

Dependency parsing is the act of analyzing the grammatical structure of a sentence and establishing relationships between "head" words and words which modify those heads, which are called "dependents". For example, in the sentence "He is my father", there exists a dependency named "nsubj" starting from the word "is" and pointing to the word "He", denoting the subject of the sentence.

Semantic role labeling is a task in NLP consisting of the detection of the semantic arguments associated with the predicate or verb of a sentence and their classification into their specific roles. For example, given a sentence like "Mary sold the book to John", the task would be to recognize the verb "to sell" as representing the predicate, "Mary" as representing the agent, "the book" as representing the theme and "John" as representing the recipient.

2.2.1 Existing Approaches

The task of open information extraction was introduced by [1] along with a system called TextRunner. Many Open IE systems have been proposed since then.

TextRunner is the first domain-independent Open IE system and it consists of three key modules. On the first one, it utilizes a parser and few heuristics to label a small corpus sample of extractions as "trustworthy" or not. For each parsed sentence, the system finds all base noun phrases and for each pair of them, the system aims to locate a potential predicate between the

two noun phrases. Then, it trains a Naive Bayes (NB) classifier in order to label future candidate extractions as "trustworthy" or not. On the second module, with the help of part-of-speech tagging and noun-phrase chunking, the system passes over the entire corpus to extract triples for all possible predicates, sends each candidate to the classifier, and retains the ones labeled as trustworthy. Finally, the last module assigns a probability to each retained triple based on a probabilistic model.

An extension of TextRunner has also been proposed [3].The latter version, in order to produce training examples, applies a set of heuristics to label positive and negative examples from the Penn Treebank. Those heuristics rely on parse trees and semantic role labeling. Features that can be extracted from the labeled examples (without syntactic or semantic analysis) are used to train a Conditional Random Field (CRF) component. For each document, TextRunner applies a phrase chunker and treats the identified noun phrases as candidate entities for extraction. Each pair of entities matching some criteria is considered as possible evidence for relation extraction. Finally, it trains the CRF to extract relation phrases. It should be noted that implicit predicates that could be inferred from the text are not obtained.

The key idea underlying **WOE** [7] is the construction of training examples by matching Wikipedia infobox values and corresponding text. These examples are used to generate an unlexicalized, relation-independent extractor. At first, a preprocessor converts the raw Wikipedia text into a sequence of sentences, attaches NLP annotations and builds synonym sets for key entities. Then, training data is created by heuristically matching attribute-value pairs from Wikipedia articles containing infoboxes with corresponding sentences in the (Wikipedia) article. Two kinds of extractors are learnt, WOEparse and WOEpos. WOEparse uses features from dependency-parse trees while WOEpos is limited to shallow features like POS tags. WOEparse uses a pattern learner to classify whether the shortest dependency path between two noun phrases indicates a semantic predicate. In contrast, WOEpos trains a CRF to output certain text between noun phrases when the text denotes such a predicate.

ReVerb [8] uses shallow syntactic processing and attempts to solve the problem of incoherent and uninformative extractions by making use of syntactical and lexical constraints. For example, considering the sentence "Faust made a deal with the devil." the system considers the relation <Faust, made, a deal> as uninformative, so it extracts <Faust, made a deal with, the devil>. Instead of extracting entities first, it extracts verbal predicate sequences based on a limited set of POS patterns. Afterwards, the entities are identified around the predicate sequence, so that the system only extracts predicate tokens between two entities tokens. Finally, REVERB outputs a confidence value for every relation.

PATTY [9] is a large resource of relational patterns that are arranged in a semantically meaningful taxonomy, along with entity-pair instances. PATTY first applies a parser to the individual sentences of the corpus to obtain dependency paths. It then detects mentions of named entities in the parsed corpus. Whenever two named entities appear in the same sentence, it extracts a textual pattern. For this purpose, it traverses the dependency graph to get the shortest path that connects the two entities. For example, given the sentence "Winehouse effortlessly performed her song Rehab." the corresponding textual pattern is "Amy Winehouse effortlessly performed Rehab". In order to capture only relations that refer to subject-relation-object triples, it only considers the shortest paths that start with subject-like dependencies (nsubj, rcmod and partmod). The final result is a WordNet-style taxonomy of binary relations.

The **SONEX** [10] system works by identifying entities and extracting predicates among them from the blogosphere. It assigns meaningful labels for each relation, enabling the construction of detailed social networks. The system consists of three distinct steps: identifying entity pairs, clustering such pairs, and labelling the resulting clusters. An entity pair is defined as the union of all sentences containing a given pair of named entities (e.g. Barack Obama and Michelle O-bama). The context of an entity pair consists of selected terms that appear between the entities in the entity pair. The clustering step groups together entities with similar context; once this is done, each cluster is analyzed and a common label is assigned by inspecting the contexts of the pairs within the cluster.

OLLIE [4] intends to automatically create a large training set which could be able to encapsulate the multiple ways in which information is expressed in text. Therefore, it utilizes pattern templates which are learned automatically from a large training set that is bootstrapped from high confidence extractions from ReVerb. OLLIE's next step is to learn open pattern templates that encode various ways of expressing relations (the arguments and the exact relation phrase). Then, these open patterns are used to extract binary relations from sentences. It first matches the open patterns with the dependency parse of the sentence and identifies the base nodes for arguments and predicates. The relations are expanded in order to convey all the information relevant to the extraction. Finally, OLLIE outputs a confidence value for every relation triple.

TreeKernel [11] is a system that proposes an unlexicalized tree structure for Open IE. Initially, the sentences are parsed and entity pairs within a certain distance are extracted. Then, it extracts textual fragments which matches certain POS patterns in an entity pair's context as predicate candidates for that pair. A predicate candidate can consist of words before, between, after the pair or combination of those. The shortest path between the two entities along with the shortest path between predicate words and an entity are used as input to the tree kernel. With kernel-

based SVMs, both learning and classification relies on the inner-product between instances, so as to avoid extracting explicit features from parse trees. The SVM is employed to decide if the relation triples are correct or not. In addition to the supervised model, the authors propose an unsupervised model which relies on several heuristic rules instead of the SVM.

ClausIE [12] defines clause as a part of a sentence that expresses some coherent piece of information; it consists of one subject, one verb, and optionally of an indirect object, a direct object, a complement and one or more adverbials. For each sentence, ClausIE uses a dependency parser to parse the sentence. Afterwards, it maps the dependency relations produced to clause constituents. Additionally, ClausIE creates a number of clauses that do not directly appear in the sentence, e.g. for the sentence "Bell, a telecommunication company, which is based in Los Angeles, makes and distributes electronic, computer and building products", the system obtains the clause (S: Bell, V: "is", O: company). Once clauses have been obtained, ClausIE tries to identify the type of each clause according to the grammatical function of its constituents. The last step is to decide which constituents form a proposition (extracted triple) and then generate the proposition from the constituents. The system operates without any post-processing and requires no training data.

EXEMPLAR [2] is a rule-based system derived from a study of all dependency types identified by the Stanford parser. The relations extracted are n-ary but can also be converted to binary. The initial step involves the use of a parser in order to identify for each sentence the dependencies between the tokens and the named entities. Then, it detects the triggers of the predicates that can be either verbs or nouns. Each trigger enables the system to also identify the predicate that includes the trigger. After predicate triggers are identified, EXEMPLAR proceeds to detect their candidate arguments and assigns each argument with a role, such as "subject", "direct object" etc. The final step consists of converting n-ary relations into binary ones. This is done by selecting all pairs of arguments from each n-ary relation and creating a new binary relation.

SwiRL [6] is based on a novel, customizable IE paradigm that takes advantage of predicate-argument structures identified automatically, which rely on extended set of features and inductive decision tree learning. In their studies, the authors found that inductive learning through decision trees enabled them to easily test large sets of features and study the impact of each feature on a parser that outputs predicate argument structures. For that reason, they used a decision tree learning algorithm to implement both the classifier that identifies argument constituents and the classifier that labels arguments with their roles.

Lund [13] is a semantic role labeling system for English that is integrated with a dependency parser. To tackle the problem of joint syntactic-semantic analysis, the system relies on a syntactic and a semantic subcomponent. The syntactic model is a projective parser using pseudo-projective transformations and the semantic model uses global inference mechanisms on top of a pipeline of classifiers. The complete syntactic-semantic output is selected from a candidate pool generated by the subsystems.

	Annotation with:			**Predicates with:**		**Method:**	
	shallow parsing	**dependency parsing**	**semantic parsing**	**nouns**	**verbs**	**machine learning**	**rule-based**
SwiRL	×	✓	✓	×	✓	✓	✓
Lund	×	✓	✓	✓	✓	✓	✓
TextRunner [NB]	✓	×	×	×	✓	✓	✓
TextRunner [CRF]	✓	×	×	×	✓	✓	✓
WOEpos	✓	×	×	×	✓	✓	✓
WOEparse	✓	✓	×	×	✓	✓	✓
ReVerb	✓	×	×	×	✓	✓	×
PATTY	✓	✓	×	✓	✓	×	✓
SONEX	✓	×	×	✓	✓	✓	×
OLLIE	✓	✓	×	✓	✓	✓	✓
TreeKernel [SVM]	✓	✓	×	✓	✓	✓	×
TreeKernel [heuristics]	✓	✓	×	✓	✓	×	✓
ClausIE	✓	✓	×	✓	✓	×	✓
EXEMPLAR	✓	✓	×	✓	✓	×	✓

Table 2.1: Grouping of Open IE systems' features

In Table 2.1 the main features of the methods described above are depicted.

Those systems can be grouped either by annotation parsing technique, by whether or not their relations are able to identify noun predicates and by the general method of the system, i.e. whether it is based on rules, machine learning or both. Most of the systems combine machine learning and rules to identify the relations, however, those approaches differ with each other. Both Swirl and Lund rely on SRL and are trained on large datasets but their overall approach differs since Swirl is based on decision tree learning whereas Lund is based on a pipeline of classifiers. TextRunner with CRF shares many similarities with WOE since they both train a CRF but differ since WOE utilizes information from Wikipedia to create its training dataset. TextRunner with CRF annotates its own training dataset using heuristics. OLLIE resembles no similarities to any of the aforementioned systems since it utilizes automatically learned pattern templates to identify relations. Although many systems have been proposed, CombIE follows different approaches regarding the merging of hand-written rules and machine learning.

A machine learning method that has been used widely is CRF. Since that method is utilized in this thesis as well, it is explained in the next chapter in more detail.

2.2.2 Open Relation Extraction and Conditional Random Fields

In machine learning and statistics, classification is the problem of identifying to which set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known. Classification is considered an instance of supervised learning, which is the machine learning task of inferring a function from labeled training data.

Structured prediction is a term for supervised machine learning techniques that involves predicting structured objects, rather than scalar, discrete or real values. Probabilistic graphical models form a large class of structured prediction models. In particular, random fields are popularly used to solve structured prediction problems in a wide variety of application domains including bioinformatics, natural language processing and speech recognition.

An example of applied structured prediction is the sequence labeling problem. In POS tagging, for instance, each word in a sequence must receive a tag that expresses the "type" of word. RE can be treated as a sequence labeling problem.

A Conditional Random Field [14] is a form of undirected graphical model that defines a single log-linear distribution over label sequences given a particular observation sequence. CRF is used

for performing labeling and segmentation tasks in order to identify the most likely sequence of labels for the words in any given sentence. It is a model that defines a conditional probability P(y|x) over label sequences given a particular observation sequence x. Conditional models are used to label a novel observation sequence x* by selecting the label sequence y* that maximizes the conditional probability P(y*|x*).

The type of CRF that is most commonly used is the linear-chain CRF. In a linear-chain CRF, the set of output label variables are arranged in a sequence. The only dependency relationships are between an output variable and the previous variable in the sequence, as well as the observations corresponding to that variable. Figure 2.1 depicts the form of a linear-chain CRF.

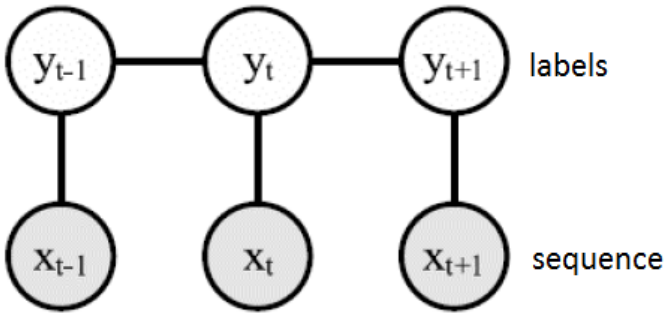

Figure 2.1: The form of linear-chain CRF

CRF has been applied to various Open RE systems [1,7]. In order to train a CRF, a set of labels and a set of features must be specified. The features describe the corresponding token sequence and may contain POS tags, punctuation etc. Figure 2.2 depicts an example from TextRunner, where "ENT", "O", "B-REL" and "I-REL" are the CRF's labels and "…Kafka, a writer born in Prague…" are the tokens of the sequence.

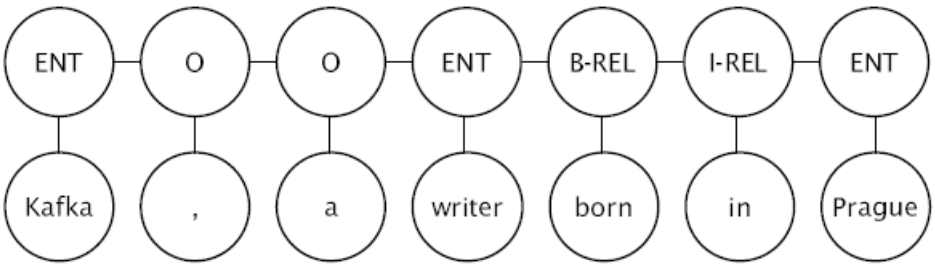

Figure 2.2: Relation extraction as sequence labeling: A CRF is used for relation extraction

2.3 Problem Statement

Among the research that has been done in Open Relation Extraction, it is a common fact that there exist more than one definitions of what is considered a relation and it mostly depends on

the researcher's point of view. For example, given the sentence "Faust made a deal with the devil" there are systems [2, 8] that consider the predicate "made a deal with" more informative and thus more preferable than "made", whereas other systems [1, 7] consider otherwise. Furthermore, some systems capture predicates consisting of verb phrases [1, 7] and others capture predicates consisting of both verbs and nouns [2, 8]. Not only predicate forms may vary but also their arguments. Arguments can be noun phrases [12] or named entities [2]. It becomes obvious that a relation may take many forms which makes it very challenging to actually capture all of them.

Taking those challenges under consideration, in this thesis a new system that attempts to solve them by combining hand-crafted rules and machine learning is proposed.

Chapter 3

Methodology

This chapter presents CombIE, a new open relation extraction system. CombIE consists of two components, the Rule-based System and the Conditional Random Fields module, whose combination produces two different versions of CombIE. The components and the two versions of CombIE are going to be explained in this chapter. As mentioned in the previous chapter, the existing Open RE systems have faced various difficulties regarding the definition of the relation and the diversity of its arguments and predicate. This system attempts to address those issues by using a combination of machine learning and hand-crafted rules in order to capture as many relation forms as possible.

3.1 Rule-based System

The system described in this thesis mostly retrieves predicates consisting of verb phrases. A few relations with noun predicates can be retrieved as well. As far as the arguments of the relations are concerned, they can be almost everything – from named entities and noun phrases to adverbs and complete sentences.

This component consists of three different functions, the Candidate Predicate Finder, the Candidate Relation Finder and the Final Relation Filtering, as shown in Figure 3.1.

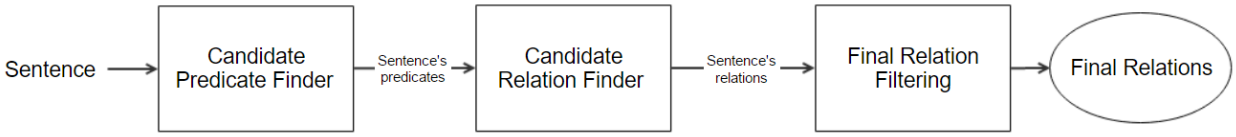

Figure 3.1: The Rule-based System

Since the Rules require NLP tasks in order to perform its processes, a library called “spaCy” was deployed. SpaCy[1] is an open-source NLP library written in Cython. Some of its functionalities include Tokenization, POS tagging, Dependency parsing and Entity Recognition.

[1] https://spacy.io/

Candidate Predicate Finder

With the assistance of spaCy, this function performs tokenization and POS tagging on each sentence. POS tagging is the process of marking up a word in a corpus as corresponding to a particular part of speech, such as nouns, verbs, adjectives, adverbs, etc. The purpose of the POS tagging in the function is to identify the candidate predicates that emerge from each sentence. Table 3.1 describes the POS tag set that was used in the thesis.

Tag	Meaning	Examples
VERB	verb	be, will, given, told
ADV	adverb	really, already, still, early, now
PART	particle	to, up
ADP	adposition	on, of, at, with, by, into, under
PRON	pronoun	he, their, her, its, my
DET	determiner	the, a, some, most
NOUN	noun	year, home, Athens

Table 3.1: Universal POS tag set

Initially, the predicate pattern was created to match the following POS tag sequence:

*ADV(negation only)[0-1] VERB+ (PART | VERB | ADP | ADV(negation only))**

According to this pattern, a predicate may have a negation in its beginning, end or in the middle of it. The reason why only the adverbs that suggest negation are included in the predicate is that most of the times they do not contain useful information when they do exist within a predicate.

A few examples of the predicates that this pattern can capture are the following:

- "I(PRON) should(VERB) have(VERB) gone(VERB) to(ADP) the(DET) party(NOUN)." where the predicate is <should have gone to>.
- "I(PRON) will(VERB) not(ADV) visit(VERB) her(PRON)." where the predicate is <will not visit>.

- "I(PRON) stepped(VERB) out(ADP) of(ADP) the(DET) house(NOUN)." where the predicate is <stepped out of>.
- "He(PRON) is(VERB) living(VERB) up(PART) to(ADP) the(DET) expectations(NOUN)." where the predicate is <living up to>.
- "I(PRON) decided(VERB) to(PART) go(VERB) to(ADP) the(DET) party(NOUN)." where the predicate is <decided to go to>.

However, in the case of the last sentence, if "to the party" was absent, according to the predicate pattern the relation would not have an object and no relation would be returned. Therefore, in order to avoid such cases the predicate pattern was converted to the following:

ADV(negation only)[0-1] VERB+ (PART | ADP | (ADV(negation only) VERB))**

According to this pattern, if an "ADP" POS tag connecting two sequences of verbs exists, then the second sequence should become part of the object. Therefore, while using the updated predicate pattern, the sentences

"I decided to go to the party."

and

"I decided to go."

would have the same predicate, which is "decided".

Candidate Relation Finder

In this function, dependency trees are utilized in order to find the arguments of each relation, if the corresponding dependencies exist.

However, since the dependent words that indicate subjects or objects most of the time lack important information (e.g. "daughter" instead of "David's daughter"), they are being replaced with the word's subtree that spaCy produced during the dependency parsing.

After extensive research, a list of dependencies that depict a link between a word in a predicate and a subject was created. A list of dependencies that depict a link between a predicate and an object was created as well. Below, these lists are presented:

Relation Objects:

- '**dobj**': When present, this dependency denotes the link between a verb in a predicate and one of its accusative objects.

For example, the dependency parse tree of the sentence “We can take the opportunity to move forward.” is the following:

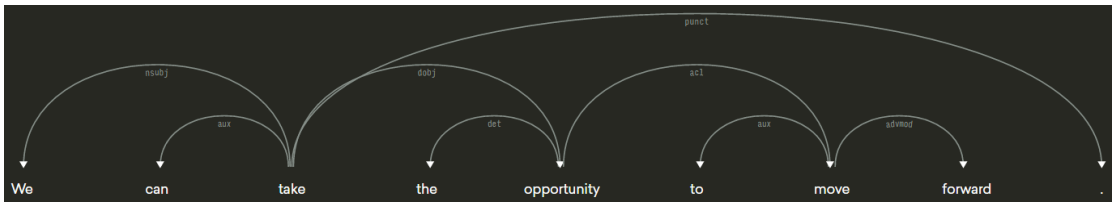

Figure 3.2: Dependency tree with the ‘dobj’ dependency

The word that the dependency ‘dobj’ is pointing at is “opportunity” and the subtree containing that word is “the opportunity to move forward”. As a result, we have the following relation:

Subject = “We”

Predicate = “can take”

Object = “the opportunity to move forward”

- ‘**pobj**’: This dependency signifies a link between a preposition in a predicate and its object. For example, in the sentence “The events occurred in July.” the predicate found is “occurred” and there exists a dependency “obj” whose head word is “in” and dependent word is “July”. In this case, the relation is the following:

  Subject = “The events”

  Predicate = “occurred in”

  Object = “July”

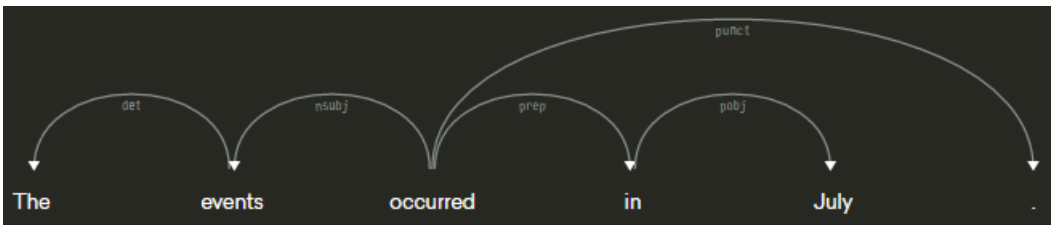

Figure 3.3: Dependency tree with the ‘pobj’ dependency

- ‘**dative**’: When this dependency links a word from a predicate to an object, most of the times there is another dependency indicating object that starts from the same predicate and points to a different word. The dependent word of the ‘dative’ dependency is usually a noun. For example, the dependency parse tree of the sentence “Jake decides to give Melanie a bit of a hard time.” is the following:

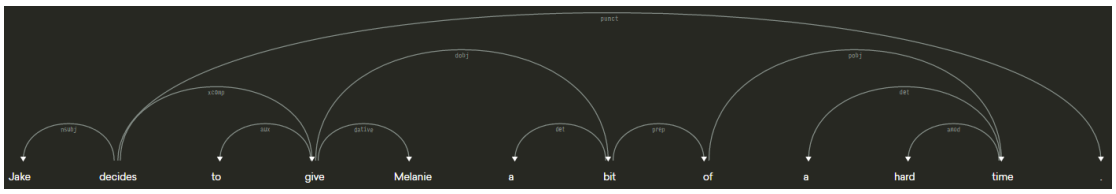

Figure 3.4: Dependency tree with the 'dative' dependency

In this case, "decides to give" is the predicate, "Melanie" is the first object deriving from the "dative" dependency and "a bit of a hard time" is the second object. The final relation is:

Subject = "Jake"

Predicate = "decides to give"

Object1 = "Melanie"

Object2 = "a bit of a hard time"

- '**attr**': This dependency links a predicate containing verbs like 'be/seem/appear' and their objects. For example, in the sentence "He became a Texan", the relation is the following:

  Subject = "He"

  Predicate = "became"

  Object = "a Texan"

- '**oprd**': This dependency points from a verb inside a predicate to an object that is either noun or adjective.
- '**acomp**': It indicates a link between a verb and an adjective complement. For example, given the sentence "The spokesperson was not available for comment." the relation is the following:
  Subject = "the spokesperson"

  Predicate = "was not"

  Object = "available for comment"

- '**ccomp**': This dependency suggests a link between a verb and a complementizer phrase. For the sentence "I am having my bathtubs reglazed." the relation is the following:

  Subject = "I"

  Predicate = "am having"

  Object = "my bathtubs reglazed"

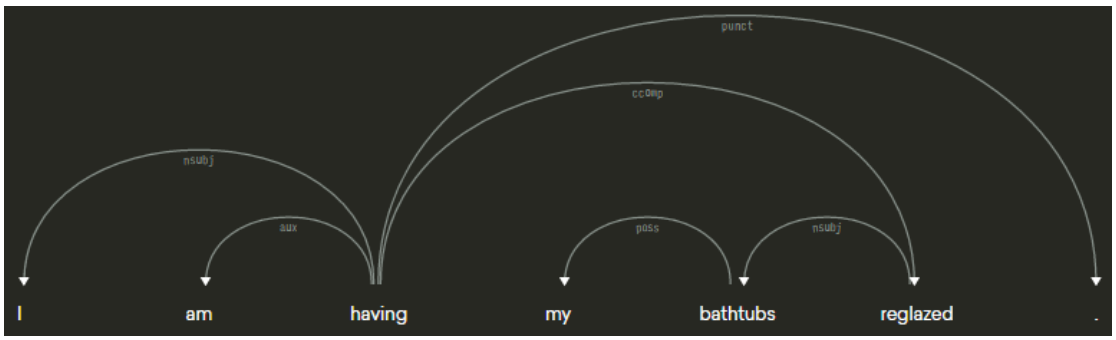

Figure 3.5: Dependency tree with the 'ccomp' dependency

- '**xcomp**': It links a verb within a predicate with a verb phrase complement outside of the predicate.

- '**advmod**', '**npadvmod**': These dependencies create a link from a word to an adverb or an adverb noun phrase. Given the sentence "Lieutenant Commander Pavlic received the Purple Heart posthumously.", the relation is the following:

  Subject = "Lieutenant Commander Pavlic"

  Predicate = "received"

  Object1 = "the Purple Heart"

  Object2 = "posthumously"

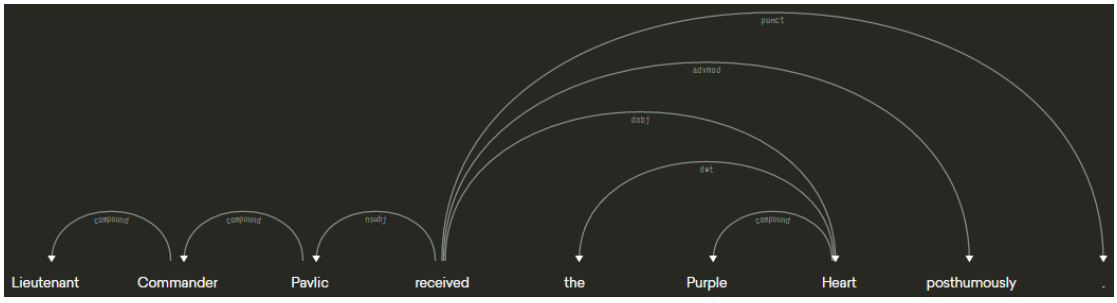

Figure 3.6: Dependency tree with the 'advmod' dependency

Relation subjects:

- '**nsubj**': This dependency indicates a link between a verb and a noun phrase subject.
  For example, for the sentence "Girard has a bachelor's degree in political sciences." the resulting relation is:

  Subject = "Girard"

  Predicate = "has"

  Object = "a bachelor's degree in political sciences"

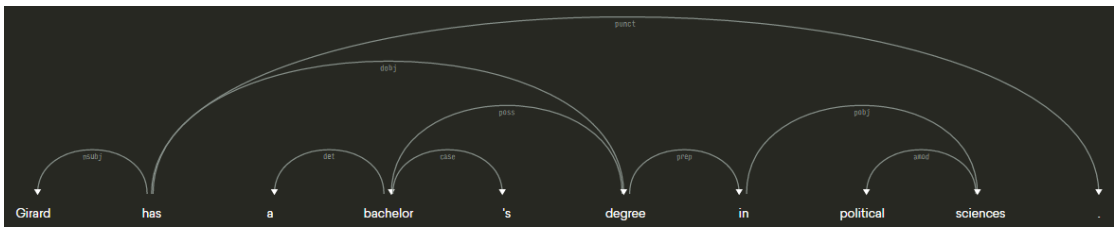

Figure 3.7: Dependency tree with the 'nsubj' dependency

- '**nsubjpass**': It signifies a link between a passive participle and a noun phrase surface subject. For the sentence "Mrs. Yogeswaran was shot five times", the resulting relation is:

  Subject = "Mrs. Yogeswaran"

  Predicate = "was shot"

  Object = "five times"

- '**csubj**': This dependency creates a link between a verb and a complementizer phrase.

Words indicating subjects that are not on the left of the predicate and words indicating objects that are not on the right side of the predicate are being omitted, since in the case of this system's rules most of the times those relations lack logical meaning.

Given the fact that each argument is being replaced by its subtree, there exists the risk that the subtree may contain inadequate or excessive information. Most of the times, the subtrees do not contain information that could be considered redundant. Some other times, the information given by them could be considered excessive but not necessarily wrong. Since there was not a golden standard of when and where those subtrees should be trimmed, they were left as produced by the parser.

The only case where the subject's dependent subtree is not considered to be the relation's subject is when the word in question is either the relative pronoun "which" or "who". In that case, it would be incorrect to accept those words as subjects, since they are referring to another person or object. In order to solve this issue, a dependency called "relcl" is utilized whose dependent's subtree is the logically correct subject.

In order to better explain the above issue, the following example is considered. Given the sentence "Lawson was the sole black member of the Homebrew Computer Club, a group of early computer hobbyists which would produce a number of industry legends, including Apple founders Steve Jobs and Steve Wozniak.", the original relations are the following:

Relation #1:
Subject = “Lawson”
Predicate = “was”
Object = “the sole black member of the Homebrew Computer Club , a group of early computer hobbyists which would produce a number of industry legends , including Apple founders Steve Jobs and Steve Wozniak”

Relation #2:

Subject = “which”
Predicate = “would produce”
Object = “a number of industry legends, including Apple founders Steve Jobs and Steve Wozniak”

Considering the object of the 1st relation, the information contained is undoubtedly extensive. Is it, however, necessarily incorrect? Given the fact that in relation extraction most of the times what is considered correct is ambiguous, the correctness of it depends on the reviewer’s judgment.

As far as the 2nd relation is concerned, its subject belongs in the list of relative pronouns that need to be replaced by the subject they are referring to. In that case, “which” is replaced by “a group of early computer hobbyists”.

Final Relation Filtering

As mentioned above, adverbs in a predicate primarily do not contain useful information. As far as objects are concerned, occasionally one of the dependencies point to a word that is an adverb or an adverb phrase. Most of the times, those adverbs contain additional information that could be avoided. In cases where there is at least another object, the dependent derived from one of the two aforementioned dependencies are removed. Otherwise, if the only existing dependency denoting object is one of them, it is kept because it is considered essential.

For instance, given the sentence “The sequel followed in 1944, one year later.”, the dependency tree is the following:

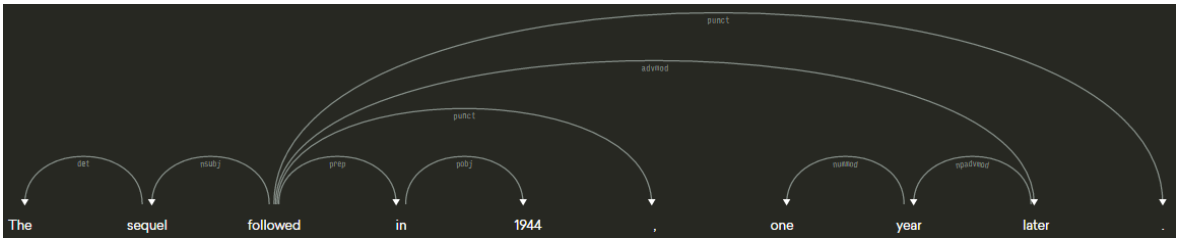

Figure 3.8: Dependency tree for the sentence “The sequel followed in 1944, one year later.”

From this sentence two objects emerge, “1944” which is the dependent of the dependency pobj and “one year later” which is the dependent of advmod. According to the above-mentioned limitations, the advmod’s dependent is discarded. If, however, the sentence was “The sequel followed one year later”, the advmod’s dependent “one year later” would not be discarded since it contains the only piece of information available.

Since in the scope of the thesis the relations are binary, if at least one subject and at least one object were not retrieved for each relation, the relation would be ignored completely. Occasionally, a relation has more than one subject. For example, given the sentence “Ronald Squire (25 March 1886 - 16 November 1958) was an English character actor.”, the resulting subjects are “Ronald Squire” and “(25 March 1886 - 16 November 1958)”, both resulting from the dependency nsubj. In order not to omit possibly valuable information, the system performs the merging of these subjects into one, resulting to “Ronald Squire (25 March 1886 - 16 November 1958)” in the specific example. The same methodology is followed for the objects of the relations as well.

3.1.1 Rules identifying Named Entities

The initial objective of the rules was to capture subjects and objects no matter what their content was. However, the dataset [see Chapter 4] that was used for the evaluation of CombIE was limited to having only named entities as subjects and objects and the extracted relations of each sentence could be one at maximum. Consequently, a few modifications had to be made to the Rules component, in order to be able to evaluate it in the used datasets.

Candidate Predicate Finder for Named Entities

Since the system is looking for relations whose arguments are necessarily named entities, there is no longer the need to use the modified predicate pattern. As a result, the predicate pattern was converted to the initial one:

*ADV(negation only)[0-1] VERB+ (PART | VERB | ADP | ADV(negation only))**

Candidate Relation Finder for Names Entities

A new rule was built in order to capture noun predicates of the following pattern:

Subject = named entity

Predicate = NOUN ADP

Object = named entity

where there exists a dependency called “appos” whose head is the subject and its dependent is the predicate’s noun, which describes the subject’s trait. For instance, given the sentence “Michele, sister of Tracy, wears a red dress.”, the noun phrase “sister of” is considered to be a predicate between “Michele” and “Tracy”.

Final Relation Filtering for Named Entities

Given the fact that according to the dataset each sentence should return one relation at most, this function had to be altered in order to satisfy that restriction.

The following algorithm describes how the final extracted relations were selected for each sentence:

Algorithm:

Input:

verb_rel: The relations with verbal predicates of the sentence. If a sentence does not return at least one relation, verb_rel is empty

noun_rel: The relations with noun predicates of the sentence. If a sentence does not return a relation, noun_rel is empty

named_entities: The named entities that exist in the sentence

limit: a limit of words that if a subject or object exceeds, the relation is deleted

Output:

rel_final: the extracted relations, maximum one relation for each sentence

For each sentence:

if noun_rel != empty *and* noun_rel[subject] *contains one of* named_entities *and* noun_rel[object] *contains one of* named_entities **then**

rel_final = noun_rel

else if verb_rel != empty **then**

for each relation:

if verb_rel[relation][subject] *contains one of* named_entities *and*
verb_rel[relation][object] *contains one of* named_entities *and*
word_count(verb_rel[relation][subject]) <= limit *and*
word_count(verb_rel[relation][object]) <= limit **then**

```
                    subject = first_named_entity_of(verb_rel[subject])
                    relation = verb_rel[relation]
                    object = first_named_entity_of(verb_extr[object])
                    extr_temp.add[subject, relation, object]
        extr_final = extr_temp with minimum sum of words in subject and object
```

According to this algorithm, if the rule that identifies noun predicates returns a relation, it is selected as the only relation of the sentence and the verbal rules are ignored. If the rules that identify verbal predicates return one or more relation, each of them is being examined in order to reject the relations that do not meet certain criteria. One criterion includes the length of the arguments in words, since an argument with many words most likely does not refer to a specific named entity. It is not mandatory for each sentence to have a relation.

3.2 Conditional Random Fields

In this section, open relation extraction is treated as a structured classification problem. The method that was selected to help with the relation extraction is CRF and the library used that implements the CRFs is called python-CRFsuite[2].

CRFsuite[3] is an implementation of Conditional Random Fields for labeling sequential data and python-CRFsuite is a python library that offers a binding to the CRFsuite.

During the initialization of the CRF, the model's parameters (e.g. training algorithm, regularization parameter) need to be specified. The features and the labels of the CRF need to be specified as well.

As far as the features that were used to train the CRF are concerned, many were taken into consideration:

- **Universal POS tag set** (e.g. VERB, ADV)
- **Penn Treebank POS tag sets**[4] (e.g. VB, NNS, RB)

[2] https://python-crfsuite.readthedocs.io/en/latest/

[3] http://www.chokkan.org/software/crfsuite/

[4] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

- **Window size of the method's conjunction of features**: possible values between the range of 2 and 6 were examined.

Two features that were not tested and were included by default is whether or not the token is a named entity and the label of each named entity (e.g. ORG, PER), if one exists.

The CRF's parameters that were considered are the following:

- **algorithm**: the list of training algorithms include 'Gradient descent using the L-BFGS method', 'Stochastic Gradient Descent with L2 regularization term', 'Averaged Perceptron', 'Passive Aggressive' and 'Adaptive Regularization of Weight Vector'
- **all_possible_transitions**: This parameter may be True of False. When True, CRF generates transition features that associate all of possible label pairs, even those that do not occur in the training data.
- **all_possible_states**: This parameter may be True of False. When True, CRFsuite generates state features that associate all of possible combinations between attributes and labels, even those that do not occur in the training data.
- **coefficient_for_L1_regularization** and **coefficient_for_L2_regularization** values for the algorithms that support them.

The labels of each token are determined as follows:

- **Trigger**: Each sentence of the development set contains the correct predicate, if one exists, and the main word of the predicate, the trigger. That word is given the "Trigger" label.
- **Entity**: The subject and the object of the relation are declared by the "Entity" label. Each sentence is labeled with two Entities regardless of the existence of a trigger word.
- **O**: If a word is not labeled as Trigger or Entity, it is labeled as O.

3.3 Merging the Extracted Relations

Since both the Rule-based system and the Conditional Random Fields components are able to produce relations, the issue of merging their results arose. The CRF classifier is able to function properly when the arguments are named entities, so the merged version combines it with the version of the Rule-based system with the named entities.

3.3.1 CombIE with Meta-Classifier

The first version of the system, called "CombIE with Meta-Classifier" is presented in Figure 3.9.

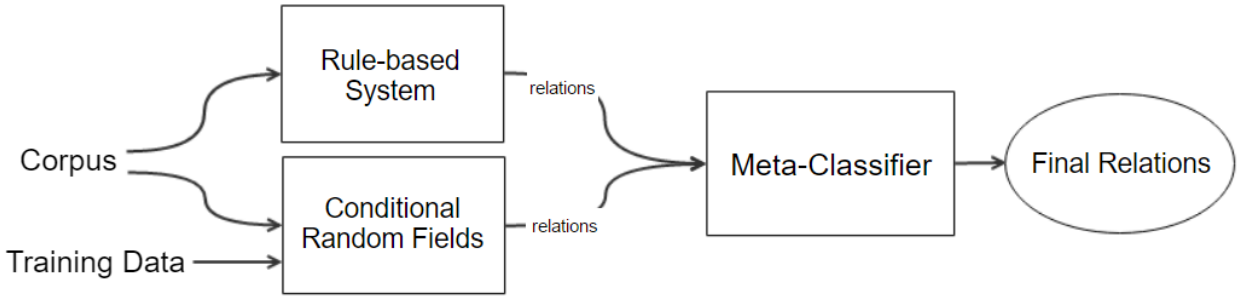

Figure 3.9: The architecture of "CombIE with Meta-Classifier"

The merging approaches that were considered and tested are listed below.

Algorithm 1:

Input:

extr_rules: Extracted relations from the Rule-based system, maximum one for each sentence. If a sentence would not return a relation, extr_rules[sentence] would be empty

extr_crf: Extracted relations from the CRF. If a sentence would not extract a trigger, extr_crf[sentence] would be empty

Output:

extr_final: merged extracted relations, maximum one relation for each sentence

For each sentence:

if extr_rules[sentence] != empty and extr_crf[sentence] == empty **then**

extr_final.add(extr_rules[sentence])

else if extr_rules[sentence] == empty and extr_crf[sentence] != empty **then**

extr_final.add(extr_crf[sentence])

else if extr_rules[sentence] != empty and extr_crf[sentence] != empty **then**

case 1: extr_final.add(extr_rules[sentence])

case 2: extr_final.add(extr_crf[sentence])

case 3: **if** extr_rules[sentence]== extr_crf[sentence] **then**

extr_final.add(extr_rules[sentence])

According to this algorithm, if the Rule-based system returns a relation and CRF returns no relation for a sentence, the final relation of the sentence is the one from the Rules. If CRF returns a relation and the Rule-based system returns no relation, the final relation of the sentence is the

one from the CRF. If both the Rule-based system and CRF return a relation, the following three approaches were considered:

- The relation that is returned is the one from the Rule-based system
- The relation that is returned is the one from CRF
- A relation is returned only if the Rule-based system and CRF have extracted the same relation

The second approach involved more extensive mingling of the two components' extracted relations. That algorithm is described below.

Algorithm 2:

Input:

extr_rules: Extracted relations from the Rule-based system, maximum one for each sentence. If a sentence would not return a relation, extr_rules[sentence] would be empty

extr_crf: Extracted relations from the CRF. Not matter if a sentence would extract a trigger or not, the relation is returned.

Output:

extr_final: merged relations, maximum one relation for each sentence

For each sentence:

```
if extr_rules[sentence] != empty then
    if count_Named_Entities(extr_crf[sentence]) == 2 then
        extr_final.add(get_Named_Entities(extr_crf[sentence]))
        extr_final.add(get_Relation(extr_rules[sentence]))
    else
        extr_final.add(extr_rules[sentence])
else if extr_rules[sentence] == empty and count_Named_Entities(extr_crf[sentence]) == 2 and count_Triggers(extr_crf[sentence]) == 1 then
    extr_final.add(extr_crf[sentence])
```

According to this algorithm, if the Rule-based system returns a relation and CRF returns two arguments, the final relation of the sentence is a combination of Rules' predicate and CRF's arguments. If the Rule-based system returns a relation and CRF does not return two arguments,

the final relation of the sentence is the one from the Rules. If the Rule-based system returns no relation and CRF does, if CRF's extraction contains exactly one trigger and two arguments, the final relation of the sentence is the one from the CRF.

The last version of merging the extracted relations from the Rule-based system and the CRF component was a modification of the Algorithm 2, attempting to decrease the false positive extracted relations. The algorithm is described below.

Algorithm 3:

Input:

extr_rules: Extracted relations from the Rule-based system, maximum one for each sentence. If a sentence would not return a relation, extr_rules[sentence] would be empty

extr_crf: Extracted relations from the CRF. Not matter if a sentence would extract a trigger or not, the relation is returned.

Output:

extr_final: merged relations, maximum one relation for each sentence

For each sentence:

```
if extr_rules[sentence] != empty and count_Triggers(extr_crf[sentence]) != 1 then
    if count_Named_Entities(extr_crf[sentence]) == 2 then
        if extr_rules[sentence]== extr_crf[sentence] then
            extr_final.add(extr_rules[sentence])

    else
        extr_final.add(extr_rules[sentence])
else if extr_rules[sentence] == empty and count_Named_Entities(extr_crf[sentence]) == 2 and
count_Triggers(extr_crf[sentence]) == 1 then
    extr_final.add(extr_crf[sentence])
else if extr_rules[sentence] != empty and count_Named_Entities(extr_crf[sentence]) == 2 and
count_Triggers(extr_crf[sentence]) == 1 then
    if extr_rules[sentence]== extr_crf[sentence] then
            extr_final.add(extr_rules[sentence])
```

According to this algorithm, in order to determine a relation, the relations from the Rule-based system and the CRF must match, otherwise no final relation of the sentence is returned.

3.3.2 CombIE with Semi-Supervised Classifier

The approach of building a semi-supervised classifier was considered as well. In Section 3.1.1 it was described how the Rule-based system functions when it is identifying named entities. CombIE with Semi-Supervised Classifier attempts to identify named entities as well.

The second version of the system, called “CombIE with Semi-Supervised Classifier” is presented in Figure 3.10.

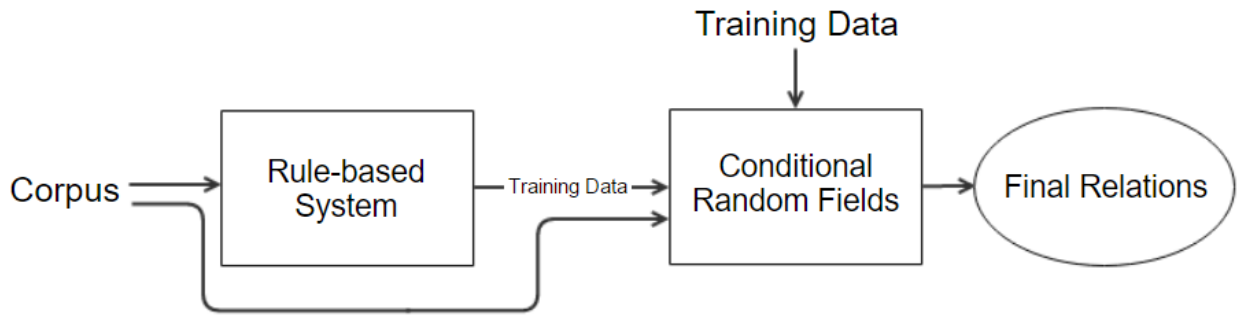

Figure 3.10: The architecture of "CombIE with Semi-Supervised Classifier"

In this version of CombIE, the Rule-based system takes as input a corpus for which the relations are sought and produces the corresponding relations. Considering those relations as correct, the Rule-based system feeds the Structured Classifier component with those relations. Afterwards, the Structured Classifier takes as input the same corpus as the Rule-based system and trains the CRF not only with the development set described in Section 4 but also with the extracted relations from the Rule-based system. The final relations that emerge are a combination of rules and classification without the need of another merging function since the merging is performed automatically by the CRF.

Chapter 4

Evaluation

In order to be able to evaluate the methods that were proposed in this thesis, experimental evaluation was performed. CombIE was compared to other Open RE systems on three different datasets. In this chapter, the datasets that were used for the evaluation of CombIE and the metrics used for the evaluation are presented. The tuning of the CRF and the experimental results are discussed as well.

4.1 Datasets

Three datasets has been used for the evaluation and were selected due to the fact that different open relation extraction systems were evaluated on them [2].

Each sentence of those datasets was annotated manually. For each one, two named entities and a trigger for the predicate between them, if one existed, were identified. In addition, a window of words allowed to be included in each predicate was specified. Those words could be modifiers of the trigger and prepositions connecting triggers to their arguments.

In the annotated sentences, entities are enclosed in triple square brackets, triggers are enclosed in triple brackets and the window of allowed tokens is defined by arrows ("--->" and "<---"), as in the following example:

I've got a media call about [[[ORG Google]]] --->'s {{{acquisition}}} of<--- [[[ORG YouTube]]] --->today<---.

where "Google" and "YouTube" are entities of the type 'organization', "acquisition" is the trigger and the allowed tokens in the predicate are "acquisition", "'s" and "of".

In particular, the selected datasets are:

- WEB-500: It is a dataset that was developed for the TextRunner experiments [3] and was annotated manually by the authors of EXEMPLAR. These sentences were selected at random from an IE training corpus and are often incomplete and grammatically unsound, representing the challenges of dealing with web text. Most of the predicates in this dataset have a verb as their trigger. 460 of its sentences were annotated with a trigger, whereas the rest 40 were considered not to contain one.

- NYT-500: It consists of sentences from the New York Times which were annotated by the authors of EXEMPLAR as well. These sentences are formal, well written and most of their triggers consist of nouns. However, the sentences are inconsistent in terms of how the predicates were formatted. In other words, the triggers of this dataset were sometimes not positioned between the named entities, e.g. "[[[ORG Prudhoe Bay]]], [[[LOC North America]]] ---> 's largest {{{oilfield}}} <---" . Only 150 sentences out of the 500 were considered to have a trigger and were annotated with it.
- PENN-100: It contains sentences from the Penn Treebank and the relations were manually annotated by the TreeKernel's authors [11]. Almost all of the predicates' triggers are nouns. An aspect of this dataset that should be noted is that all of the named entities' labels are described as 'NONE' whereas the other two datasets contain the ['MISC', 'LOC', 'PER', 'ORG', 'NONE'] labels. 52 sentences were annotated with a trigger.

A corpus of 300 annotated New York Time sentences that was available along with the other datasets of EXEMPLAR was used as development and training set for the CRF classifier of CombIE. Only 92 sentences of it were annotated with triggers. All the datasets are shown on table 4.1.

Dataset	# of Sentences	# of Relations
WEB-500	500	460
NYT-500	500	150
PENN-100	100	52
NYT-300 (development set)	300	92

Table 4.1: The datasets

It is worth noting that there exist some contrasts between the annotations of WEB-500 and NYT-500 and the annotations of PENN-100. For instance, one difference concerns the definition of an entity, since in the PENN-100 dataset sometimes consider words nearby a named entity as part of a named entity while in the other two datasets this is not the case. Another difference lies in the fact that the PENN-100 dataset was annotated without the need for the systems to perform co-referencing, which was later altered by the authors of EXEMPLAR. These differences demonstrate the challenges of producing a benchmark for open relation extraction.

Finally, it should be taken into account that all of the aforementioned datasets were annotated considering that the ORE systems are responsible for resolving co-references.

4.2 Evaluation Metrics

The metrics used for the evaluation are precision (P), recall (R) and f-measure (F_1), and are defined as follows:

$$P = \frac{\#\ of\ correct\ extractions\ output\ by\ the\ system}{\#\ of\ total\ extractions\ output\ by\ the\ system}$$

$$R = \frac{\#\ of\ correct\ extractions\ output\ by\ the\ system}{\#\ of\ relations\ in\ dataset}$$

$$F_1 = \frac{2 * P * R}{P + R}$$

4.3 Experiments

For each sentence, an extracted relation is considered correct if the system extracts the correct annotated entities, the string representing the trigger and the allowed tokens of the predicate, if they exist. In addition, there was made the assumption that there is only one predicate between a pair of entities in a sentence.

The authors of EXEMPLAR, in order to perform a fair evaluation regardless of how well each system's parser was able to perform, replaced the named entity that denotes each sentence's subject with the word "Europe" and the named entity that denotes the object with the word "Asia". The reason for the replacement lies in the fact that they empirically found that for 99.7% of the sentences in their experiments, all methods were able to recognize "Europe" and "Asia" as entities and there was no conflict with the pre-existing entities of the sentences. For methods that extract noun phrases [4, 6, 8, 13], they performed the additional task of identifying whether a noun phrase containing additional words surrounding "Europe" and "Asia" is still a reference to the annotated entity. They ignored completely noun phrases that did not reference the annotated entity.

As far as the merging algorithm that was chosen for CombIE with Meta-Classifier is concerned, the algorithms were examined by using the development set. The Algorithm 2 produced better results than any or the variations of the Algorithm 1. The Algorithm 3 did decrease the false

positive extracted relations but also decreased the false negatives, so the Algorithm 2 was the selected one to be executed in CombIE with Meta-Classifier.

4.3.1 Conditional Random Field Tuning

During the preprocessing phase of the CRF, each sentence of the development set is parsed and all the named entities are replaced with a single word. This step is performed since a named entity may consist of more than one tokens and it would be simpler to identify the named entity as a whole rather that sequential tokens.

In order to extract the features and the CRF's parameters that optimize the results, 3-fold cross validation was performed on the development set.

The selected features were the following:

- **POS tags**: Neither the Penn Treebank's POS tags nor the combination of the Universal POS tags and the Penn's Treebank POS tags proved to be more effective than the Universal ones.
- **Window Size = 3**: While all windows between 2 and 6 were tested, the most effective one was 3 for a span of tokens' features to the left and 3 tokens' features to the right.

The CRF's attributes that optimize the results are the following:

- **algorithm** = The Passive Aggressive algorithm[5] was chosen for the training of the model.
- **all_possible_transitions** = True
- **all_possible_states** = True

A Boolean value that denotes whether or not the token is a named entity and the label of each named entity (e.g. ORG, PER), if one exists, were added to the aforementioned feature list.

The CRF attempts to extract the named entities and the trigger of each sentence. The rest of the predicate's tokens that were annotated in each sentence could not be indentified due to the small size of the training set, so they were ignored completely in the training phase of the CRF. Since the identification of those tokens was optional, the evaluation among all systems was fair.

[5]http://scikitlearn.org/stable/modules/generated/sklearn.linear_model.PassiveAggressiveClassifier.html

An example of a training sentence is the following:

Original sentence:

"Kris Knapton is a spokesman for Metra, a commuter rail service in northeastern Illinois."

where "Metra" is the subject, "Illinois" is the subject and "service" is the trigger**.**

Sentence after replacing the named entities with words that imply the existence of named entities:

"namedentity0 is a spokesman for namedentity1, a commuter rail service in northeastern namedentity2."

The labels of the sentence:

namedentity0→ 'O'	namedentity1→ 'Entity'	service→ 'Trigger'
is→ 'O'	, → 'O'	in→ 'O'
a→ 'O'	a→ 'O'	northeastern→ 'O'
spokesman→ 'O'	commuter→ 'O'	namedentity2→ 'Entity'
for→ 'O'	rail→ 'O'	. → 'O

The features of namedentity1:

{'-3entity': False, '-2entity': False, '-1entity': False, 'entity': True, '+1entity': False, '+2entity': False, '+3entity': False, '-3postag': 'DET', '-2postag': 'NOUN', '-1postag': 'ADP', '+1postag': 'PUNCT', '+2postag': 'DET', '+3postag': 'NOUN', 'label': 'MISC'}

All the feature names starting with a negative number indicate words' features on the left of the word "namedentity1" and all the features starting with a positive number indicate words' features on the right. Since the word "namedentity1" is a named entity, there is no POS tag of it in the features. The features also include the label of the specific named entity.

In order to reduce the false positive results from CRF, a function that calculates the confidence of each relation was written. Since the CRFSuite measures the confidence for each label that it returns, the average confidence for the returned labels that were either 'Entity' or 'Trigger' was

calculated. However, those confidence scores proved to be uninformative so this function was not utilized during the filtering.

4.3.2 Experimental Results

Table 1 presents the results of the evaluation. For each column, the best results are marked in bold. The evaluation of the state of the art systems was performed by [2]. They were unable to run TreeKernel on the NYT-500 and WEB-500 datasets due to lack of training data. They ran TreeKernel, as trained by its authors, on the same test set used in their paper [11]. EXEMPLAR[M] and EXEMPLAR[S] describe two versions of the same system, where the first one implements the Malt parser and the second one implements the Stanford parser.

The dataset that the systems performed their best is WEB-500, which is evident by their f-measure score when tested in it. The average recall of all systems is also higher than the other two, probably due to the fact that that it mainly contains verbs as triggers and all of the systems were built able to identify verb predicates. Another reason could be that those sentences were collected by querying a search engine with known relation instances. Although it was expected that WEB-500 would represent the challenges found in web text, that was not the case.

The other two datasets, NYT-500 and PENN-100, contain randomly chosen sentences and most of their triggers are nouns. Systems that do not recognize noun predicates did not perform that well. Reverb and Swirl recognize only verb predicates, which is evident by their low recall. SONEX and Lund, which are respectively similar to Reverb and Swirl but identify noun predicates, managed to achieve higher recall.

Both versions of EXEMPLAR, which are rule-based and cover a wide range of both verb and noun predicates, performed really well.

As far as CombIE is concerned, both its two components and the two different versions that resulted from their combination were evaluated separately. The results of their evaluation are described below.

Method	NYT-500			WEB-500			PENN-100		
	P	R	F1	P	R	F1	P	R	F1
ReVerb	0.70	0.11	0.18	0.92	0.29	0.44	0.78	0.14	0.23
SONEX	0.77	0.22	0.34	**0.98**	0.30	0.45	**0.92**	0.43	0.59
OLLIE	0.62	0.27	0.38	0.81	0.29	0.43	0.81	0.43	0.56
EXEMPLAR[M]	0.70	**0.39**	0.50	0.95	0.44	0.61	0.83	0.49	**0.62**
EXEMPLAR[S]	0.73	**0.39**	**0.51**	0.96	0.46	**0.62**	0.79	**0.51**	**0.62**
PATTY	0.49	0.23	0.32	0.71	**0.48**	0.57	0.46	0.24	0.31
SwiRL	0.63	0.10	0.17	0.97	0.34	0.50	0.89	0.16	0.27
Lund	**0.78**	0.24	0.37	0.91	0.37	0.52	0.86	0.35	0.50
TreeKernel	-	-	-	-	-	-	0.85	0.33	0.48
Rule-based System	0.45	0.29	0.36	0.64	0.37	0.47	0.53	0.37	0.43
CRF	0.57	0.16	0.25	0.62	0.09	0.16	0.07	0.02	0.03
CombIE with Meta-Classifier	0.43	0.37	0.40	0.65	0.41	0.50	0.43	0.38	0.40
CombIE with Semi-Supervised Classifier	0.49	0.24	0.32	0.78	0.47	0.58	0.40	0.35	0.37

Table 4.2: The results for the evaluation process

Evaluation for Rule-based System

In terms of recall, the main factor contributing to the Rule-based system's results is the fact that only one rule was written for noun predicates, whereas a large amount of sentences had a variety of noun predicates. The lack of noun rules might also have caused a drop in precision, given that

if a relation with a noun predicate existed in the Rule-based system, it was always considered as the sentence's relation, regardless if verb predicates existed as well.

It is also possible that since CombIE implements spaCy and other systems implement other parsers, a small variation might occur, as with EXEMPLAR[S] and EXEMPLAR[M]. Since all parsers are prone to errors in parsing, spaCy is too. Errors in POS tagging and dependency parsing also result in extraction errors. For instance, verbs with capitalized the first letter were sometimes not considered verbs but named entities by spaCy.

Moreover, since the Rule-based system extracts predicates that are between named entities, the component was unable to capture the cases where the trigger was not between the entities.

CombIE was created with a view to extracting subjects and objects that could contain whole sentences. With the modification of subjects and objects in order to consist of named entities, many extracted relations lost significant meaning, resulting to incorrect relations.

Some sentences did not output a relation or output the wrong one due to co-referencing, which was common in the evaluation datasets. CombIE resolved co-references only when the subject was relative pronoun, whereas the annotators of the datasets considered that all pronouns should be co-referenced.

The limitation that each sentence should contain one relation at maximum caused a drop both in precision and recall, since CombIE identified verb predicates between named entities that were considered incorrect by the annotators of the datasets.

In addition, there rose the most important challenge of relation extraction which is subjectivity regarding what should be considered correct or not. For instance, given the sentence:

"LEAD : The Campeau Corporation 's rocky 18-month adventure in American retailing has run into its biggest cash squeeze yet , forcing its [[[LOC American]]] ---> subsidiaries , which {{{own}}} <--- Bloomingdale 's , [[[ORG Jordan Marsh]]] and other national chains , to say they might have to file for bankruptcy protection ."

CombIE extracted the following:

Subject = "American"
Predicate = "own"
Object = "Bloomingdale 's"

which could also be considered correct.

As far as the results are concerned, given the lack of pattern in NYT-500 dataset's sentences and the fact that it consists mainly of noun triggers, the Rule-based system extracted 44 correct

relations where 27 of them had verb triggers. Regarding the Rule-based system's extracted relations that were considered incorrect by the annotators, most of those cases included entities linked by a verb that was not annotated by the annotators.

Out of the 17 correct relations extracted from the PENN-100 dataset, only 7 had verb triggers.

Given that in WEB-500 most of the triggers were verbs, it was expected that the recall would be higher compared to the other two datasets. The main reason for its wrong extracted relations was the existence of predicates that consisted of both verbs and nouns. CombIE is unable to capture predicates that contain both verbs and nouns. An example that describes the case where a predicate that contains both verbs and nouns would be correct if its object was not limited to contain a named entity is:

"[[[NONE Google]]] ---> announced Tuesday that it {{{bought}}} <--- [[[NONE YouTube]]] for $ 1.65 billion ."

The Rule-based system initially extracted the following:

Subject = "Google"
Predicate = "announced"
Object = "that it bought YouTube for $ 1.65 billion"

which after the modification of the object in order to contain only a named entity became:

Subject = "Google"
Predicate = "announced"
Object = "that it bought YouTube for $ 1.65 billion"
which lacks important information and is therefore incorrect.

Evaluation for CombIE - CRF component only

Given that the training set it really small, the annotated triggers in it are even fewer and the datasets for the evaluation do not share many similarities with each other, it was expected that CRF would not perform that well. During the testing, all of the datasets had an f-measure score in the 'named entities' label that exceeded 0.9, probably because the named entities were annotated in the datasets even when the sentences did not contain a trigger.

Although NYT-500 and the training set originated from the same corpus, CRF mostly extracted triggers from sentences that were not annotated with one. It extracted both verb and noun triggers but most of them were verbs.

Named entities' labels play an important role in identifying the correct trigger of each sentence. Without the feature of the named entities' labels, the f-measure score of the extracted relations drops significantly. Since the labels of the PENN-100 dataset's entities were all annotated as "NONE", CRF was unable to identify the correct triggers, although almost all of the named entities were correctly extracted. Even though the PENN-100 mostly had nouns as triggers, it varied from the training set. As a result, the triggers that were extracted originated from sentences that were not annotated with one.

The correct extracted relations from WEB-500 confirm that the CRF managed to learn that when a verb is surrounded at close proximity by named entities, that verb is a trigger.

Evaluation for CombIE with Meta-Classifier

The Meta-Classifier managed to successfully combine the extracted relations from the Rule-based system and the CRF component. Their combination also produced relations that otherwise would be considered incorrect. Considering the high f-measure score of the named entities originating from the CRF component, the correct extracted relations from the Rule-based system were all still correct after the replacement of their named entities by the CRF's.

In the NYT-500 dataset, the combination of an incorrect relation from the CRF and an incorrect relation from the Rule-based system produced one new correct relation. In the PENN-100 dataset it produced one correct relation and in the WEB-500 it produced nine correct relations that would otherwise be incorrect.

Evaluation for CombIE with Semi-Supervised Classifier

The results of the Semi-Supervised Classifier were unpredictable, unlike the Meta-Classifier's. In WEB-500 its performance was good, whereas in the NYT-500 and PENN-100 the performance was slightly worse that the Rule-based system's.

There was no obvious pattern for the correct extracted relations from the Rule-based system and the CRF that were not predicted correctly when combined. Some of them were completely different than their trained instances and some of them did not succeed in predicting the correct trigger or predicted more named entities and triggers. The main factor though was the inability of the Semi-Supervised Classifier to predict the trigger.

The reason behind those incorrect extracted relations could be the low precision of the extracted relations from the Rule-based system and the CRF or the fact that NYT-500 and PENN-100 have both noun and verb triggers and a conflict between them might occur. On the other hand, the

Semi-Supervised Classifier performed really well on the WEB-500 and the reason probably lies in the fact that this dataset primarily consists of verb triggers and there is no conflict between the triggers.

Moreover, this classifier produced correct relations that were considered incorrect both by the Rule-based system and the CRF. Most of these relations were extracted by the Meta-Classifier as well.

As far as NYT-500 is concerned, almost all of the extracted relations from the Rule-based system and the CRF that were both correct were also correct in the Semi-Supervised Classifier. Only one third of the correct extracted relations from the Rule-based system were also extracted by the classifier. What is more, 4 new correct relations that the individual components were unable to detect, were extracted correctly.

While evaluating the classifier on the PENN-100 dataset, 3 correct relations that were incorrect both in CRF and in Rule-based system were extracted. A few of the correct Rule-based system's extracted relations were incorrect when extracted from the Semi-Supervised Classifier as well.

All the correct results that were extracted from WEB-500 by the Rule-based system were extracted as well by the Semi-Supervised Classifier. All of the correct results that were extracted both from the Rule-based system and the CRF were extracted as well by the classifier. All of the extracted relations from the Rule-based system and the CRF that were both correct were also correct in the Semi-Supervised Classifier. However, only half of the correct results that were extracted originally by the CRF component were extracted by the classifier as well.

4.3.3 Discussion

Considering the fact that the training dataset had such a small amount of relations, the CRF was able to predict relations correctly and therefore increase the recall in CombIE with Meta-Classifier. CombIE with Meta-Classifier was able to perform better than its two individual components in terms of recall.

As far as CombIE with Semi-Supervised Classifier is concerned, its results were completely unpredictable and probably more experiments should be conducted in order to have a better understanding of its performance.

Compared to the other state of the art systems, CombIE was not as effective. However, the results of CombIE with Meta-Classifier in terms of recall are promising. It is not clear as to if CombIE with Semi-Supervised Classifier could be promising as well.

Chapter 5

Conclusion and Future Work

In order to solve the problems that emerged from previous Open RE systems, a new system called CombIE was proposed in this thesis. CombIE consists of two components, a Rule-based system and a CRF module. Each of those components is able to extract relations which are then combined in two different ways to produce CombIE with Meta-Classifier and CombIE with Semi-Supervised Classifier.

Experimental evaluation of the two versions of CombIE and their individual components was performed and the overall performance of Meta-Classifier and CombIE was better than Semi-Supervised Classifier's. It is possible that by training CRF on a larger dataset the recall of both systems could improve noticeably.

A step for the improvement of CombIE could be a function that effectively manages to detect relations that are not considered correct, so as to increase the systems' precision. What is more, the Rule-based system could be expanded by rules that extract predicates which contain both verbs and nouns.

ABBREVIATIONS

IE	Information Extraction
NLP	Natural Language Processing
RE	Relation Extraction
SRL	Semantic Role Labeling
POS	Part-of-speech
NB	Naive Bayes
CRF	Conditional Random Field

BIBLIOGRAPHY

[1] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni, “Open Information Extraction from the Web”. Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, pp. 2670-2676, 2007.

[2] F. Mesquita, J. Schmidek and D. Barbosa, “Effectiveness and Efficiency of Open Relation Extraction”. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, pp. 447 - 457, 2013.

[3] M. Banko and O. Etzioni, “The Tradeoffs Between Open and Traditional Relation Extraction”. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, Columbus, Ohio, USA, pp. 28-36, 2008.

[4] Mausam, M. Schmitz, S. Soderland, R. Bart and O. Etzioni, “Open Language Learning for Information Extraction”. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, pp. 523-534, 2012.

[6] M. Surdeanu, S. M. Harabagiu, J. Williams and P. Aarseth, “Using Predicate-Argument Structures for Information Extraction”. Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, pp. 8-15, 2003.

[7] F. Wu and D. S. Weld, “Open Information Extraction Using Wikipedia”. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, pp. 118-127, 2010.

[8] A. Fader, S. Soderland and O. Etzioni, “Identifying Relations for Open Information Extraction”. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, pp. 1535-1545, 2011.

[9] N. Nakashole, G. Weikum and F. M. Suchanek, “PATTY: a Taxonomy of Relational Patterns with Semantic Types”. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, pp. 1135-1145, 2012.

[10] Y. Merhav, F. de Mesquita, D. Barbosa, W. G. Yee and O. Frieder, “Extracting information networks from the blogosphere”. TWEB, 6(3): 11:1-11:33, 2012.

[11] Y. Xu, M. Y. Kim, K. Quinn, R. Goebel and D. Barbosa, “Open Information Extraction with Tree Kernels”. Conference of the North American Chapter of the Association of Computational Linguistics, Atlanta, Georgia, USA, pp. 868-877, 2013

[12] L. Del Corro and R. Gemulla, “ClausIE: clause-based open information extraction”. 22nd International World Wide Web Conference, Rio de Janeiro, Brazil, pp. 355-366, 2013

[13] R. Johansson, P. Nugues, “Dependency-based Semantic Role Labeling of PropBank”. Proceedings of the Conference on Empirical Methods in Natural Language Processing, Hawaii, USA, pp. 69-78, 2008.

[14] J. D. Lafferty, A. McCallum and F. C. N. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. Proceedings of the

Eighteenth International Conference on Machine Learning, Williamstown, USA, pp. 282-289, 2001