Unsupervised Lexicon Acquisition for HPSG-Based Relation Extraction

Benjamin Rozenfeld and Ronen Feldman
Digital Trowel and Hebrew University of Jerusalem
Israel
grurgrur@gmail.com, ronen.feldman@huji.ac.il

Abstract

The paper describes a method of relation extraction, which is based on parsing the input text using a combination of a generic HPSG-based grammar and a highly focused domain- and relation-specific lexicon. We also show a method of unsupervised acquisition of such a lexicon from a large unlabeled corpus. Together, the methods introduce a novel approach to the "Open IE" task, which is superior in accuracy and in quality of relation identification to the existing approaches.

1 Introduction

Relation Extraction (RE) is the task of recognizing instances of specific relationships between two or more entities in a natural language text. In a traditional setting, the target relation types are known to a RE system in advance, and it can be prepared for its task either by a knowledge engineer hand-crafting the extraction rules, or by the system itself learning the rules from a set of hand-labeled training examples. Both ways require a large expenditure of manual labor.

In recent years, [Banko and Etzioni 2008] introduced a new setting for the RE task, called Open Information Extraction (OpenIE). In this setting, the RE system does not know the target relations in advance, and cannot have any relation-specific human input. Thus, the task requires the system itself to identify the target relations and to train itself for extracting them.

IE systems that work on free text can perform either a shallow or a deep parsing. The advantages of shallow parsing are high speed, simplicity of training and usage, and consequent higher accuracy. So, most of the state-of-the-art IE systems, including the OpenIE systems cited above, do not use deep parsing, because its obvious theoretical advantages are offset by the practical limitations of existing general-purpose parsers.

In this paper, we describe an RE system that does perform deep parsing, using a parser which is built specifically for the task of Information Extraction. The parser's underlying framework, called CARE-II, is capable of parsing arbitrary weighted typed-feature-structure-based context free grammars. This gives the framework a unique power, allowing it to use a high-level unification-based grammar, such as HPSG, while still being able to flexibly interface with feature-rich sequence classifiers, such as CRF-based NER and PoS taggers [McCallum and Li 2003; Rosenfeld, Fresko et al. 2005; Avinesh and Karthik 2007], and to produce for each sentence a single best parse (among the exponentially many possibilities that can be allowed by a grammar), as evaluated according to the weights of the grammar and the classification scores produced by the sequence classifiers.

The parser we built on top of the CARE-II framework is generic – it uses an HPSG-like grammar (derived primarily from the example grammar in [Sag, Wasow et al. 2003]), but does not have lexical definitions for any words beside the most frequent functional (non-content) ones. Instead, it relies on the CRF-trained NER and PoS sequence classifiers to provide weights for different possible typed-feature-structure assignments for different words. And then, for any input sentence, the parser is able to generate a single highest-weight parse – the parse which is the most consistent with the NER and POS classifiers.

The resulting parses contain many errors, and would probably not compare favorably with the results of the best existing standalone parsers. However, the goal of the system is not to provide a general-purpose parser, but to be easily adaptable for the IE task. And, given a target relation type, the system only requires the definitions for a small number of the relevant content words in order to function as an accurate parser for the *relevant* sentences – the ones that actually contain instances of the target relations.

These definitions (the domain-specific lexicon) can be written manually. It is much easier than manually creating a set of extraction patterns or manually labeling a training set of sufficient size for automatic learning. However, this is still not suitable for the OpenIE task, where the target relations are not known beforehand. Thus, in this paper we describe a method for automatically extracting the required lexical entries from a large unlabeled text corpus.

2 CARE-II Framework

CARE-II is a framework for building Information Extraction systems that focus on natural language sentences. It includes a grammar description language and the supporting tools. The core of the framework is a parser, which is capa-

ble of parsing arbitrary weighted typed-feature-structure context-free grammars (WTFSCFG-s).

2.1 WTFSCFG

WTFSCFG is a WCFG (weighted context-free grammar), in which any symbol, terminal or non-terminal, carries a typed feature structure (FS); the grammar rules have access to the feature structures of their component symbols, building from them the feature structures of their heads, using the operations of unification, slot extraction, and slot removal.

The full syntax of typed feature structures in CARE-II grammar language is rather complicated, so in this paper we will use the standard notation of attribute-value matrices, with an additional '=' operator, which can be used in the feature structures attached to the heads of grammar rules. For example, the Head-Specifier Rule [Sag, Wasow et al. 2003] can be written as follows ("Phrase" is a nonterminal symbol of a grammar):

$$Phrase\boxed{2}\left[\text{SYN}\left[\text{VAL}\left[\text{SPR} = \langle\ \rangle\right]\right]\right]\ :-\ \langle -1.0\rangle$$
$$Phrase\boxed{1}\ Phrase\boxed{2}\left[\text{SYN}\left[\text{VAL}\begin{bmatrix}\text{SPR} & \langle\boxed{1}\rangle\\ \text{COMPS} & \langle\ \rangle\end{bmatrix}\right]\right];$$

The "<–1.0>" specifies a constant cost associated with the application of the rule, the *weight* part in the WCFG. [Note: all general rules have the same small constant cost, to encourage the compactness of a parse.] In other respects, the rule is a simple CFG production rule

Phrase :– *Phrase Phrase*;

extended with the feature structures. The CFG rule states that two adjacent *Phrase*-s can be reduced to a single *Phrase*. And the feature structures in the full WTFSCFG rule above constrain the FS-s that can be carried by these *Phrase*-s in the following ways: (1) the value of SYN.VAL.COMPS of the FS carried by the second *Phrase* in the rule body must be unified with an empty list, (2) the value of SYN.VAL.SPR of the same FS must be unified with a single-element list, (3) this single element must be unified with the FS carried by the first *Phrase* in the rule body, and (4) the resulting *Phrase* will carry the second feature structure, exactly as it becomes after all the unifications and after further setting the value of SYN.VAL.SPR to empty list. (Note: this last operation removes the direct results of the unification of SYN.VAL.SPR with $\boxed{1}$, but keeps the changes that resulted from co-referencing values within the feature structure.)

2.2 Terminal symbols

The CARE-II framework works at the level of tokens (words, punctuation, etc), so each terminal symbol of a grammar must span zero or more tokens. Like non-terminals defined by grammar rules, the terminal symbols can also have weights and may carry feature structures. The ambiguities are allowed – the same words can be matched to different terminal symbols, with associated different weights and feature structures.

The framework allows several varieties of terminals: direct tokens, tokens defined by regular expressions, word lists and sequence classifier labels. For the purposes of the grammar in this paper, only two kinds are relevant: first, there is a single word list, called *Lexicon*, which lists all explicitly defined words in the grammar, together with their weights and feature structures. And second, there are sequence classifier labels, described below.

2.3 Sequence Classifiers

In general, a sequence classifier is a component that takes as input a sequence of tokens (the input sentence) and selects for each token a single label from a small predefined set of labels. For example, a named entity recognizer (NER) would select the labels from a set like {"Person", "Organization", "Location", "None"}, while a part-of-speech tagger (PoS) would select the labels from a set like {"NN", "NNP", "JJ", "VB", "VBG", …}.

The CARE-II framework is able to use any sequence classifier that normally works by assigning (in whatever way) weights to different labels and/or pairs of labels at every position in the input sentence, and then selecting the best (highest-scoring) sequence of labels using the Vitterbi algorithm. This includes start-of-the-art feature-rich CRF-based or MaxMargin-based NER and PoS taggers [Rosenfeld, Fresko et al. 2005]. The classifiers are not utilized in the standard way, however. Instead of first running a classifier and then using the produced single fixed sequence of labels, the framework retrieves the actual weights that the classifier assigns, and uses them directly as the weights of the terminal symbols. So, instead of the classifier's Vitterbi algorithm, it is the more general WTFSCFG inference algorithm, which finds the highest-scoring grammar *parse*, which maximizes the sum of the weights of words, rules, and classifier labels that participate in it.

This method of using sequence classifiers makes the framework very flexible, since it allows the labels of individual tokens to change from the labeling that would be given by standalone NER/PoS, if the change is perceived to be beneficial to the whole parse. This, in turn, allows a very general, only partially lexicalized grammar, which contains rules for only the most frequent but not *all* linguistic constructions, to still achieve reasonable parsing quality. And it allows to seamlessly extend the grammar with domain-specific lexical entries in order to get high-quality parsing for sentences in the specific domain. This last trait is very important in a parsing framework used for information extraction.

2.4 Parsing Algorithm

The parsing algorithm is an extension of the Agenda-based parser for PCFG-s [Klein and Manning 2001]. The extension allows arbitrary weights, as well as feature structures and their manipulation. In the general case, inference with feature structures is Turing-machine-powerful, so the algorithm's worst-case space and time complexity is exponen-

tial. However, with careful writing of the grammar rules and feature structures, it can be made to stay efficiently polynomial.

2.5 Generic Grammar

The English grammar that is used for this paper is based on HPSG, as described in [Sag, Wasow et al. 2003] and adapted for the CARE-II framework. It contains around thirty general rules and the lexicon definitions for several hundred functional words: determiners, pronouns, prepositions, auxiliary verbs, etc.

The content words are defined generically, with the help of PoS and NER sequence classifiers, CRF-trained using the training corpora from CoNLL-2000 [Tjong, Sang et al. 2000] and CoNLL-2003 [Tjong, Sang et al. 2003] shared tasks, respectively. The definitions allow any word in the input sentence to be assigned to any word class (in the HPSG sense), with the weights set as specified by the classifiers.

The grammar's two main non-terminals are *Phrase* and *Word*. The *Phrase* rules are the generic HPSG-like rules, which build more complex phrases from simpler phrases and from *Word*-s, in the manner similar to the Head-Specifier rule shown in the section 2.1 above. *Word* functions as the single terminal symbol for these rules, but itself is defined by several rules, which connect it to the true terminals. There are three kinds of these rules. First, there is a simple rule:

$$Word\ \boxed{1} :- Lexicon\ \boxed{1};$$

which allows any word defined in the *Lexicon* word set to function as a *Word*. Then, there is a set of rules for generic content words that are not named entities. For example, the rule for generic adjectives looks like this (slightly simplified):

$$Word\left[\begin{array}{ll} \text{SYN} & \left[\begin{array}{ll} \text{HEAD} & adj \\ \text{VAL} & \left[\begin{array}{ll} \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \\ \text{LMOD} & \left[\begin{array}{ll} \text{SYN} & \left[\begin{array}{ll} \text{HEAD} & noun \\ \text{VAL.SPR} & \langle * \rangle \end{array}\right] \\ \text{SEM} & \left[\text{INDEX}\ \boxed{1}\right] \end{array}\right] \\ \text{RMOD} & \langle\ \rangle \end{array}\right] \\ \text{GAP} & \langle\ \rangle \end{array}\right] \\ \text{SEM} & \left[\text{RESTR}\ \left\langle\left[\begin{array}{ll} \text{RELN} & adj \\ \text{ARG} & \boxed{1} \end{array}\right]\right\rangle\right] \end{array}\right]$$

$$:- JJ + None;$$

Here, *JJ +None* means "any word, with the weight equal to the weight assigned by the PoS classifier to 'JJ' and NER classifier to 'None'". Similar rules are written for nouns (NN*), adverbs (RB*), and various verb forms (VB*). The verb forms have three separate definitions each, for intransitive, transitive, and ditransitive forms. There is no need to define separate generic forms for verbs with preposition phrase (PP) complements, since they can be parsed as PP modifiers. The verbs with non-standard complements (raising and control verbs, complementation verbs) are directly listed in the lexicon together with the functional verbs, but without non-generic semantic information.

Finally, there is a set of rules for named entities: Person, Organization, and Location. For example, the Person rule is:

$$Word\left[\begin{array}{ll} \text{SYN} & \left[\begin{array}{ll} \text{HEAD} & \left[\begin{array}{ll} noun & \\ \text{FORM} & nform\_Person \\ \text{AGR} & \left[\text{PER}\quad third\right] \end{array}\right] \\ \text{VAL} & \left[\begin{array}{ll} \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \\ \text{LMOD} & \langle\ \rangle \\ \text{RMOD} & \langle\ \rangle \end{array}\right] \\ \text{GAP} & \langle\ \rangle \end{array}\right] \\ \text{SEM} & \left[\text{RESTR}\quad \langle\left[\text{RELN}\quad Person\right]\rangle\right] \end{array}\right]$$

$$:- NNP + Person;$$

The "semantics" part of the HPSG grammar is integrated with the CARE-II in such a way as to produce a parse in the form of a special labeling of the input sentence. In this labeling, every sequence of tokens that corresponds to a *Word* inside the parse is put under a tag. The tags have unique ID attributes that correspond to SEM.INDEX values. The other tag attributes correspond to subslots of SEM.RESTR. The grammar is built in such a way that *Words* in a parse never intersect, and so the tags never intersect also, and the parses can be interpreted as a word dependency graphs, with words having links pointing to other words.

3 Example Relation-Specific Lexicon

For an example relation, we will use the *Acquisition* relation between companies, as in the sentence:

Qualcomm has acquired Elata for 57 million in cash.

For simplicity we will consider only the two main fields: the *Acquirer* and the *Acquired*, both of the type 'Organization'. In order to allow the framework together with the generic grammar to extract the relation instance

Acquisition(*Acquirer* = "*Qualcomm*"
Acquired = "*Elata*")

from the sentence above, it is sufficient to define a single content word – the verb "to acquire". The definition can look like this:

DefVerb: <1> “acquire”,

$$\left[\begin{array}{l} \textit{stv\_lxm} \\ \\ \text{ARG\_ST}\left\langle \left[\begin{array}{ll}\text{SYN.HEAD.FORM} & \textit{nform\_Org}\\ \text{SEM.INDEX} & \boxed{1}\end{array}\right], \left[\begin{array}{ll}\text{SYN.HEAD.FORM} & \textit{nform\_Org}\\ \text{SEM.INDEX} & \boxed{2}\end{array}\right]\right\rangle; \\ \\ \text{SEM.RESTR}\left\langle \left[\begin{array}{ll}\text{RELN} & \textit{Acquire}\\ \text{ACQUIRER} & \boxed{1}\\ \text{ACQUIRED} & \boxed{2}\end{array}\right]\right\rangle \end{array}\right]$$

The additional positive weight encourages the parses involving the verb “acquire” over generically defined words.

If this definition is added to the grammar, the sentence above will be parsed into the following:

<1: RELN=Org> Qualcomm </1>
<2: RELN=perfect ARG=3> has </2>
<3: RELN=Acquire ACQUIRER=1 ACQUIRED=4 PAST=true>
acquired </3>
<4: RELN=Org> Elata </4>
…

from which it is trivial to extract the relation instance frame. Note, that the same grammar will correctly extract the relation from sentences containing the verb “acquire” in any form and tense, including gerund. The word “acquisition”, on the other hand, is not accessible to the grammar rules, and so must be defined separately. Other words that may need to be defined are: “purchase”, “shares”, “interest”, “assets”, etc. The number of relevant words is usually not too large – on the order of 10-50 per relation, with many of the words sharing definitions, and so the task of writing them is more accessible for manual knowledge engineering than writing traditional extraction rules or labeling a large training corpus. The OpenIE task, however, requires the system to generate the relations and the lexicon automatically.

4 Unsupervised Domain-Specific Lexicon Acquisition

The general idea of the acquisition method is to extract and analyze a set of frequent patterns from a large unlabeled corpus. Unlike most methods of unsupervised (or even supervised) pattern extraction, the goal here is to find patterns that are both semantically and linguistically meaningul. Only such patterns can be directly converted into lexical definitions. On the one hand, this severely restricts the possible pattern forms. But on the other hand, the lexical entries become much more general than the patterns from which they were originally generated, because the full power of the grammar’s general rules is allowed to work on the lexical entries. And yet, their extractions remain just as precise, because only linguistically correct constructions are allowed by the grammar.

The first stage of the method is to parse a large unlabeled corpus using the generic grammar. This may result in many parsing mistakes, but genuine meaningful patterns would still appear much more frequently in the parsed text than random mistakes, which is used for their identification. In the second stage, the extracted frequent patterns are converted into lexical entries and the corpus is reparsed using the combined grammar. This produces much more precise results and also significantly increases recall. In the third stage, the relations extracted by different lexical patterns are filtered, compared to each other, and merged together into single relation types if they have sufficient overlap. In the final stage, names are given to the relation types and their slots, and the lexical entries are re-generated accordingly, producing the final domain-specific lexicon.

4.1 Patterns

The research on OpenIE [Banko and Etzioni 2008] indicates that there exist several common linguistic pattern types which produce the majority of interesting relations. All of them and more are included in the set of pattern types that are checked by our system.

The patterns are pieces of parses, and the parses are word dependency graphs. Thus, all patterns are connected dependency sub-graphs, and each one includes at least two entity placeholders. There are three sets of pattern types: verb-based, noun-based, and BE-based.

The main set of patterns is the verb-based. In these patterns the head word is a verb, and the entities are either subjects, or objects, or connected to the verb or to its object via a sequence of preposition phrases. For example:

X/Org ←*s*— acquired —*c*→ Y/Org
X/Org ←*s*— merged ←*m*— with —*c*→ Y/Org
X/Org ←*s*— completed —*c*→
→ acquisition ←*m*— of —*c*→ Y/Org

The link types *s*, *c*, and *m* indicate subject, complement, and modifier, respectively. These and several other links are defined in the generic grammar. The raising and complementation verbs are excluded from participation in patterns, so “X continues to claim that it will acquire Y” is equivalent to “X acquired Y” for the purposes of pattern extraction.

The noun-based patterns are headed by a noun, and the entities are connected via preposition phrases, possessives, or compounds. For example:

(acquisition ←*m*— of —*c*→ X/Org) ←*m*— by —*c*→ Y/Org
X/Org ←*poss*— ’s —*m*→ acquisition ←*m*— by —*c*→ Y/Org
merger ←*m*— of —*c*→ (X/Org ←*conj*— and —*conj*→ Y/Org).

Finally, the BE-patterns are headed by the verb “be” in its predicative (non-auxiliary) sense:

X/Person (is) ←— mayor ←*m*— of —*c*→ Y/Loc
X/Person , ←— mayor ←*m*— of —*c*→ Y/Loc ,
X/Org ←*poss*— ’s —*m*→ headquarters (are) ←
← in —*c*→ Y/Loc

The predicative BE is different from other verbs because of the way it behaves semantically. It does not have its own semantics, instead directly connecting the predicate to its subject. The same behavior is used for the semantics of appositives, so they are equivalent to the BE-patterns.

4.2 Initial Patterns Extraction

The patterns extraction in the first stage of the process is straightforward: every pattern that belongs to one of the types described in the previous section and that appears at least 10 times is extracted and saved.

No distinction is made at this point between patterns which are identical except for attachment of preposition phrases, because the generic grammar alone does not have enough information to make a correct informed decision about the PP attachment.

If two patterns differ only by the entity types, and if one of them is much more frequent than the other (10 times or more), then the less frequent pattern is removed, in the assumption that its appearance is the result of entity recognition errors.

4.3 Initial Lexical Entries Generation

The lexical entries are generated directly from the patterns, in a mostly straightforward way. For example, the pattern "X/Org acquired Y/Org" from the section 4.1 would generate the definition of the verb "acquire" from section 3, identical in all respects except for the relation and slot names. The names are created automatically, and are non-informative at this point in the process.

The patterns that contain preposition phrases require additional decisions before they can be converted into lexical entries: whether to declare the PP as a complement or as a modifier, and to which word should the PP be attached if there is a choice. These decisions can be made statistically, by counting the frequencies of the PP-s appearing after the corresponding verbs and nouns, and the frequencies of the same words appearing alone or with different prepositions. In our current experiments, though, a simpler method was chosen: all prepositions get connected to the pattern's head word as complements, except for two cases: (1) the preposition "of" appearing between two nouns is always interpreted as a possessive, and is considered to be equivalent to the possessive clitic "'s" and to noun compounds when one of the NP-s is an entity; and (2) if one pattern is fully contained inside another, then the PP-s from inside the embedded pattern are attached to the head of the embedded pattern, not the containing one.

4.4 Clustering of the Results

There are many possible ways to cluster the patterns and their extractions in order to produce the final relation types. Since the main focus of this work is elsewhere, we chose the simplest method: to combine any two patterns with sufficiently overlapping sets of extractions, where "sufficiently overlapping" means having at least two common relation instances.

Finally, the clustered patterns can be used to generate names for the relation and for its slots. This is also not the main focus of the work, so no sophisticated naming scheme was used. Just the main word of the most frequent pattern in a cluster, qualified if necessary (in case of ambiguities) by adding the types of the arguments and other words in the pattern.

5 Experimental Evaluation

We used two large text corpora for evaluation: the "ACMM" corpus and the RCV1 corpus of Reuters news [Lewis, Yang et al. 2004], both of which were used to evaluate the URIES system [Rosenfeld and Feldman 2007]. This allows us to compare the quality of our linguistically-aware relation identification to the one based on clustering of shallow parse patterns and their extractions.

The ACMM corpus consists of a concatenation of four separate corpora, for the *Acquisition*, *Merger*, *MayorOf*, and *CEO_Of* relations. Each sentence in the corpus contains at least one keyword related to one of these four relations. The relations identified by our system were somewhat more general: the system merged together *Acquisition* and *Merger*, and combined *CEO_Of* with other head positions in companies (e.g., "*Chairman_Of*"). In other respects, the system achieved better accuracy than its predecessors. It also identified and extracted many other relations. The partial list of extractions is shown in Table 1:

Identified Relation	Count	Precision
Acquisition/Merger (Org, Org)	38499	0.93
ExecutiveOf (Person, Org)	4350	0.96
MayorOf (Person, Loc)	690	0.97
Headquarters_In (Org, Loc)	426	0.84
Represent/Advise (Org, Org)	363	0.75
IsMemberOf (Person, Org)	190	0.85
…		

Table 1. Extractions from the ACMM corpus

The precision figures were estimated by randomly checking 100 relation instances from each cluster. The relation names are manually given.

The RCV1 corpus consists of all sentences from the news articles, without preselecting. Thus, the relations identified and extracted from it are more diverse. The top several identified relations are shown in Table 2:

Identified Relation	Count	Identified Relation	Count
Acquisition/Merger	11211	PeopleInteract	2858
TalkedWithNews	8208	AppointExecutive	1459
VisitedLocation	5094	RatingOfCompany	1245
RelationsCountries	3384	People Talk/Write	460
President/Head of	3060	SportsTeamCoach	235
LocatedIn	2982	…	

Table 1. Top identified relations in the RCV1 corpus

There is no easy way to numerically compare our system to the state-of-the-art OpenIE system TextRunner [Banko and Etzioni 2008; Lin, Etzioni et al. 2009]. However, it is easy to show specific instances of general problems – sentences for which TextRunner systematically makes mistakes, yet which are processed correctly by our system, because it is more linguistically aware and utilizes deep parsing. For example, the TextRunner query for "is mayor" produces such results as:

- "LDA" from the sentence "*The LDA is the Mayor's agency for business and jobs.*"
- "year" from "*Later this year he is elected Mayor for the first time.*"
- "bright spots" from "*Look around, you'll see bright spots in city Morganton News Herald - The following is Morganton Mayor Mel Cohen's state of the city speech.*"

These examples should demonstrate the TextRunner's weakness. Even when the query is constrained to "Person is mayor of City", which reduces the recall from several thousands to just 33 instances, the last sentence from the three above still remains.

6 Related Work and Conclusions

Sekine [2006] coined the term "on demand information extraction" and developed a system that reduced the human labor needed to create IE systems in new domains. Shinyama and Sekine's [2006] preemptive IE system utilized clustering techniques to discover relationships between binary entities in a corpus of news articles. Open IE was first introduced in [Banko and Etzioni 2008].

The CARE-II framework is a descendant of the older CARE (CRF-Assisted Relation Extraction) framework, which used the same approach of combining weighted discriminative context-free grammars with Vitterbi-based sequence classifiers, but lacked the typed feature structures. It was primarily intended for supervised IE.

There are several directions, in which our system can be further improved and extended. One is allowing the system to actively seek the sentences, which can confirm or disprove the hypotheses about whether two patterns describe the same or different relations. The system may use the instances of one pattern to generate possible candidate phrases of the other, and use a search engine to look for them in the Web. Success or failure in finding them would be evidence for equivalence or non-equivalence of the relations.

Some parts of the system are yet in preliminary stages and need further work: PP attachment discovery, relation naming.

Another important shortcoming of the system is its reliance on predefined entity types. By using statistical entity discovering systems, and then applying the patterns discovery, it is possible to extract new entity types and new relations containing them.

Also, the extracted patterns may be context-dependent, and it may significantly improve the accuracy of the system to use text clustering / text classification methods when processing huge heterogenous text corpora like Web.

References

[Avinesh and Karthik 2007] Avinesh, P. and G. Karthik. Part-Of-Speech Tagging and Chunking using Conditional Random Fields and Transformation Based Learning. *Proceedings of SPSAL-2007*.

[Banko and Etzioni 2008] Banko, M. and O. Etzioni. The Tradeoffs Between Open and Traditional Relation Extraction. *ACL-08: HLT*.

[Klein and Manning 2001] Klein, D. and C. D. Manning. An $O(n^3)$ Agenda-Based Chart Parser for Arbitrary Probabilistic Context-Free Grammars. *Technical Report, Stanford*.

[Lewis, Yang et al. 2004] Lewis, D. D., Y. Yang, et al. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5: 361-397.

[Lin, Etzioni et al. 2009] Lin, T., O. Etzioni, et al. Identifying Interesting Assertions from the Web. *CIKM'09*.

[McCallum and Li 2003] McCallum, A. and W. Li. Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. *Proceedings of CoNLL-2003, Edmonton, Canada*: 188-191.

[Rosenfeld and Feldman 2007] Rosenfeld, B. and R. Feldman. Clustering for Unsupervised Relation Identification. *CIKM'07*.

[Rosenfeld, Fresko et al. 2005] Rosenfeld, B., M. Fresko, et al. A Systematic Comparison of Feature-Rich Probabilistic Classifiers for NER Tasks. *PKDD*.

[Sag, Wasow et al. 2003] Sag, I. A., T. Wasow, et al. *Syntactic Theory: A Formal Introduction*, CSLI Publications.

[Sekine 2006] Sekine, S. On-Demand Information Extraction. *ACL-2006*.

[Shinyama and Sekine 2006] Shinyama, Y. and S. Sekine. Preemptive Information Extraction using Unrestricted Relation Discovery. *HLT-NAACL 2006*.

[Tjong, Sang et al. 2003] Tjong, E., K. Sang, et al. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition, Edmonton, Canada.

[Tjong, Sang et al. 2000] Tjong, E. F., K. Sang, et al. Introduction to the CoNLL-2000 Shared Task: Chunking. *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal.