A Two-stage Bootstrapping Algorithm for Relation Extraction

Ang Sun
Department of Computer Science
New York University
New York, NY, 10003, USA
asun@cs.nyu.edu

Abstract

Bootstrapping has been empirically proved to be a powerful method in learning lexico-syntactic patterns for extracting specific relations such as book-author and organization-headquarters. However, it is not clear how to adapt this method to extract more general relations such as the employment-organization (EMP-ORG) relation. Relations like EMP-ORG are actually a set of relations which involves many nominals such as executive, secretary, officer, editor and soldier. To address this challenge, we propose a two-stage bootstrapping algorithm in this paper. The first stage is a commonly used bootstrapping framework, starting with a small set of seeds (entity pairs) and a large corpus to learn relation patterns which are further used to extract more seeds. We combined it with a second stage bootstrapping which takes as input the relation patterns learned in the first stage and aims to learn relation nominals and their contexts. After the two-stage bootstrapping learning, we incorporate features extracted from learned nominals and their contexts into a state-of-the-art SVM based relation extractor and we observe a 2% gain in F-measure.

Keywords

Information Extraction; Relation Extraction; Two-stage Bootstrapping

1. Introduction

Relation Extraction is a challenging Information Extraction (IE) task which needs to find instances of predefined relations between pairs of entities. For example, there is an employment-organization (EMP-ORG) relation between entities *CEO* and *Microsoft* in the phrase *the CEO of Microsoft*. One way to combat this challenge is by applying machine learning techniques to a corpus with relation annotations. Supervised learning systems such as (Kambhatla 2004), (Zhou et al., 2005) and (Zhao and Grishman 2005) extract diverse lexical and syntactic features from an annotated corpus to train their system. While a supervised relation extraction system could achieve promising results, its portability to new domains is limited by the availability of annotated corpora. Porting such systems to new domains would involve substantial expert manual labor.

Another direction in addressing this challenging problem is using semi-supervised methods such as bootstrapping techniques. A bootstrapping-based system only needs a small set of seed examples and an unannotated corpus. These seeds are used to generate relation patterns, which in turn result in new examples being extracted from the corpus. For example, Brin (1998) uses bootstrapping for extracting pairs of book titles and authors from HTML documents. Agichtein and Gravano (2000) uses bootstrapping for extracting organization and location pairs which participate in the organization-headquarters relation from a large collection of plain texts. This paper characterizes these systems as single-stage bootstrapping since they carry out a loop from seeds to patterns and from patterns to seeds.

Previous research in using single-stage bootstrapping for relation extraction has been focusing on relations which are specific and do not seem to contain subtypes of relations. However, there are many other relations which are really a set of relations. Take EMP-ORG for example; it contains at least 3 different types of relations, executive-organization, staff-organization and other-organization (where the contexts are not sufficient enough to determine whether a person holds a managerial or general staff position in the organization). One can imagine that, compared to the organization-headquarters relation, there are more diverse ways of stating employment than headquarters of organizations. In particular, relation patterns for EMP-ORG involve more relation nominals including *executive*, *head*, *manager*, *programmer*, *editor* and many others. Suppose we start with the seed *Bill Gates* and *Microsoft*; a simple question for single-stage bootstrapping is how could we learn nominal patterns with *economist* or *editor*, involving words other than synonyms of the position of Bill Gates such as *CEO*, *chairman* or *head*?

To address this problem, we propose here a novel bootstrapping algorithm which we call two-stage bootstrapping[1]. The first stage is a commonly used single-stage bootstrapping learning framework, i.e. it starts with seeds to learn patterns and uses learned patterns to extract more seeds. The second stage bootstrapping takes as input the relation patterns learned from the first stage. It first picks out informative nominal patterns which are then used to generate queries for learning new nominals. For

[1] Two-stage bootstrapping framework is different from the multi-level bootstrapping used in (Riloff and Jones 1999) which uses a Meta-bootstrapping stage for evaluating learned seeds (NPs) and extracting the most reliable ones to restart an iteration of bootstrapping.

example, suppose we could learn a relation pattern *PERSON, former chairman of ORGANIZATION* in the first stage which is then selected by our algorithm as an informative nominal pattern; we generate a query *PERSON, former * of ORGANIZATION* to search for new nominals which could replace the wildcard. Newly learned nominals would be evaluated and high-confidence ones will be instantiated back into the patterns and passed to the first stage for extracting more seeds.

The next section first gives an overview of our two-stage bootstrapping learning framework, and then it briefly describes the *Snowball* system which serves as the basis for most of the components in our first stage bootstrapping. It also shows the details of our second stage bootstrapping, mainly explaining how to choose informative nominal patterns and how to evaluate new nominals. Section 3 will show our experiments using two-stage bootstrapping for learning relation nominals and contexts. In section 4, we extract features from learned nominals and contexts and incorporate them to improve a state-of-the-art SVM-based relation extraction system. Section 5 draws our conclusion and points out our future work.

2. A Two-stage Bootstrapping Algorithm for Relation Extraction

2.1 The Two-stage Bootstrapping Framework

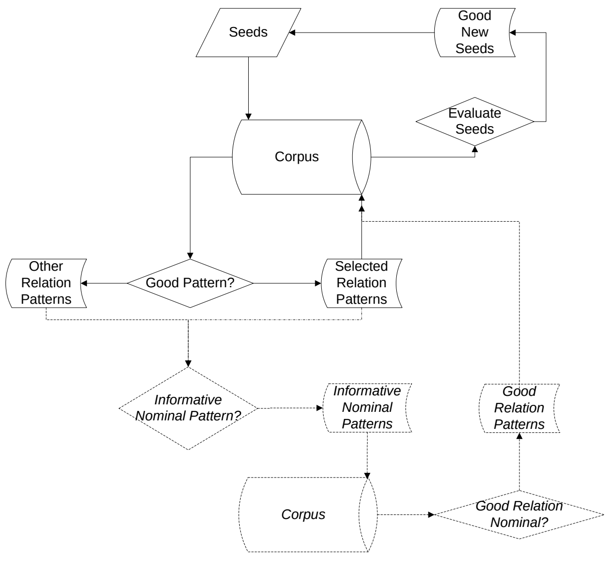

Figure 1. A two-stage bootstrapping framework

Figure 1 shows the main components of our two-stage bootstrapping learning framework. We will briefly describe the algorithm in the domain of EMP-ORG. However, it's worthy pointing out that this is a general learning framework and can be adapted to other relations. As long as a relation has a second type of evidence (relation nominals in EMP-ORG) which is associated with and could be derived and harvested from the first type of evidence in the first stage (relation patterns in EMP-ORG), the algorithm should be ready to be ported to that relation.

The two-stage bootstrapping algorithm works as follows (where the second stage is shown in *italics*):

1. Start from seeds (Person and organization pairs).

2. Search corpus for sentences containing both names.

3. Extract relation patterns from sentences.

4. Evaluate patterns:

a. Evaluate relation patterns and select high confidence ones;

b. *Select informative nominal patterns and "translate" them to relation nominal queries, e.g. PERSON, former head of ORG* $\rightarrow$ *PERSON, former *NN* of ORG.*

5. Search for new nominals using nominal queries.

*6. Evaluate new nominals, extract high-confidence ones and transform queries back to relation patterns, e.g. PERSON, former *NN* of ORG* $\rightarrow$ *PERSON, former editor of ORG* (if we learned *editor* as a high-confidence nominal).

7. Use relation patterns both from 4.a and 6 to search for new name pairs.

8. Evaluate extracted new name pairs and add the most reliable ones to the seed set.

9. If the algorithm has not reached stopping criteria, go to 2.

2.2 The First-stage Bootstrapping Framework

Our first-stage bootstrapping learning adopts most of the major components of the *Snowball* system (Agichtein and Gravano 2000).

Snowball starts with a seed set containing some initial valid seeds in the form of <*o*, *l*> such as <*Microsoft*, *Redmond*> meaning that Microsoft is an organization whose headquarters are located in Redmond. It then searches for segments of text in the text collection where *o* and *l* occur close to each other and generates patterns. A pattern in *Snowball* is a 5-tuple <*left*, *tag1*, *middle*, *tag2*, *right*>, where *tag1* and *tag2* are named-entity tags, and *left*, *middle*, and *right* are vectors associating weights with terms. For example, it generates a 5-tuple < {<*the*, *0.2*>}, *LOCATION*, {<*-*, *0.5*>, <*based*, *0.5*>}, *ORGANIZATION*, {} > for *the Redmond-based Microsoft*. The confidence of a pattern *P* is then estimated by the following formula. Good patterns should match more positive seeds than negative ones.

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

Top ranked patterns are then used to generate more seeds. A seed will have high confidence if it is generated by multiple high-confidence patterns. Good seeds are then used to start a new iteration of the bootstrapping learning.

$$Conf(seed) = 1 - \prod (1 - Conf(P_i))$$

We adopt Snowball's confidence measures for evaluating patterns and seeds in our first-stage bootstrapping. We represent a pattern by a 4-tuple <*order*, *tag1*, *middle*, *tag2*>, where in the domain of EMP-ORG, *order* means either PERSON-ORG or ORG-PERSON, *tag1* and *tag2* are named-entity tags, *middle* is the middle tokens between the two named-entities. For example, our system generates a 4-tuple <*PERSON-ORG*, *PERSON*, {*, former chairman of*}, *ORG*> for *Bill Gates, former chairman of Microsoft.*

2.3 The Second-stage Bootstrapping Framework

In this stage, bootstrapping first picks out nominal patterns from all the patterns returned by the first stage, then it selects informative nominal patterns for constructing relation nominal queries[2]. Queries are used to search for new nominals which will be evaluated and the top ranked ones will instantiate the queries to relation patterns. Those patterns are then used together with good patterns selected in the first stage to search for new name pairs.

2.3.1 Pick Out Nominal Patterns

We use a simple heuristic procedure to pick out nominal patterns. One thing we should mention is that we use all the relation patterns learned in the first stage as input to our procedure. The reason for including patterns which are not selected as good patterns in the first stage is that bootstrapping usually expands the pattern set in a very cautious way to guarantee learning quality. A good bootstrapping algorithm only adds the most reliable ones to grow its pattern set. However, patterns not being selected might be good nominal patterns. We will let the second stage decide the usefulness of these patterns.

The procedure first tokenizes and tags the middle part of each relation pattern with a HMM POS tagger. If a pattern contains tags *NN* or *NNS*, i.e. if it contains a common noun, then it is selected as a candidate nominal pattern. For example, P2 is a candidate while P1 is not.

P1: PERSON ,/, who/WP co/VBZ -/: founded/VBD ORG

P2: PERSON ,/, the/DT billionaire/NN chairman/NN of/IN ORG

Then for each common noun in each candidate, the following two heuristics are applied.

H1: if there is no common noun to its right, then it is the *head*. In P2, *chairman* is the *head* noun while *billionaire* is not.

H2: if H1 is true, check if the first modifier to the left of *head* is an article (a, the) or determiner (these, etc.); If not, annotate the candidate with *head* information. Articles and determiners are not good selective modifiers for learning new nominals so we do not annotate these kinds of patterns with *head*. For example, P3 would not be annotated with *head* information.

P3: PERSON ,/, the/DT chairman/NN of/IN ORG

The procedure then clusters 2 patterns together if they have the same *head* noun annotation. Finally for each cluster, if its size is larger than a threshold *t* (5 in the final experiment), then send it to the next procedure for selecting informative nominal patterns.

2.3.2 Select Informative Nominal Patterns

To estimate a pattern's selectivity, we compute **Bigram Mutual Information** (**BMI**) and **Dice** statistics between the head nominal and its direct modifier (the first modifier to its left). For example, we only consider BMI/Dice between *executive* and *director* in *PERSON, chief executive director of ORG*.

BMI, *MI(x;y)*, compares the probability of observing *x* and *y* together (the joint probability) with the probabilities of seeing *x* and *y* independently.

$$MI(x;y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Table 1 shows an example for the bigram *senior chairman*.

Table 1. Contingency table for x=senior y=chairman

	y	-y	
x	N_{11} = 98	N_{12} = 329,653	N_{1+} = 329,751
-x	N_{21} = 337,308	N_{22} = 1,836,348,898	N_{2+} = 1,836,686,206
	N_{+1} = 337,406	N_{+2} = 1,836,678,551	N_{++} = 1,837,015,957

[2] These are queries not to an IR or Web search engine but rather to a text search engine which searches for sequences of tokens with wildcards. The queries are generated by replacing the nominal in a pattern with a wildcard and used to search for other nominals which could replace the wildcard in the pattern. For example, we generate a query *PERSON, former * of ORGANIZATION* based on pattern *PERSON, former chairman of ORGANIZATION*. We then use it to look for nominals other than *chairman* which could replace the wildcard.

We compute all these statistics in an 86-year news corpus with 1.9 billion tokens and 1,837,015,957 bigrams not crossing sentence boundaries. N_{11} is the total number of times of observing the bigram *xy*; N_{12} is the number of times *x* occurs in bigrams to the left of words other than *y*; N_{21} is the number of times *y* occurs in bigrams after words other than *x*; and N_{22} is the number of bigrams containing neither *x* nor *y*. The probabilities can be approximated by[3]: $P(x) = N_{+1}/N_{++}$, $P(y) = N_{1+}/N_{++}$, $P(x,y) = N_{11}/N_{++}$. Then BMI can be computed as:

$$MI(x;y) = \log_2 \frac{N_{++}N_{11}}{N_{+1}N_{1+}}$$

Similarly, we can approximate *Dice(x,y)* by:

$$Dice(x,y) = \frac{2P(x,y)}{P(x)+P(y)} = \frac{2N_{11}}{N_{+1}+N_{1+}}$$

We compute BMI/Dice for each pattern in each cluster of nominal patterns and add the top ranked ones to a pattern set *S*. After the computation, we use the top ranked patterns in *S* to construct nominal queries.

2.3.3 Evaluate New Nominals

A Good nominal should be able to cross several patterns, i.e. it should match several queries. Basing a nominal's quality on one query is error-prone. So we assign a confidence score to a new nominal in the following way:

$$Conf(nom) = \sum_{i=1}^{t} \#Q$$

where *i* is the index of iterations, *t* is the maximum number of iterations during the experiment and *#Q* is the number of queries that *nom* matched during the i^{th} iteration. A nominal's confidence is updated crossing different iterations for the reason that a "loser" in the current iteration might become a "winner" in later ones. Selected new nominals are used to instantiate queries to relation patterns which will be passed to the first stage to participate in finding new name pairs.

3. Experiments for Discovering Relation Nominals

We conducted 2 experiments, one uses single-stage bootstrapping for learning relation nominals and the other one uses two-stage bootstrapping. Common parameters and tools used are summarized in Table 2.

The single-stage bootstrapping extracts all nominals from nominal patterns being picked out by the procedure described in section 2.3.1. It only learned 24 nominals, {***chairman***, *company*, ***executive***, *founder*, ***leader***, ***president***, *office*, *year*, ***director***, ***officer***, ***head***, *billionaire*, *giant*, *investor*, *group*, ***coach***, ***member***, *owner*, ***chief***, *network*, ***general***, *investment*, *opposition*, ***minister***}. One could easily judge based on common knowledge that the bold ones are correct nominals for EMP-ORG relation. All other nominals are either wrong or need context to decide.

Table 2. Common parameters and tools used

Seeds	*<Bill Gates ; Microsoft>* *<Louis Gerstner ; IBM>*
Corpus	[7]
Search engine	[7]
POS and NE tagger	Jet[4]
Maximum length of middle context	7 tokens
Maximum iteration	10

In two-stage bootstrapping, each nominal has a set of patterns matching it and each such pattern might match it several times. We assign a score to each learned nominal according to these two factors and only keep a nominal whose score is larger than 2, i.e. there are patterns/pattern matching it at least 3 times. In this way, *BMI* discovered 958 relation nominals and *Dice* 1096 nominals.

We then face the challenging problem of evaluating what we learned in the second stage. It is difficult to evaluate the results in stage two in isolation. Also, it is not feasible to directly use the ideal metric evaluation methodology suggested by *Snowball* since for one thing there is no perfect list of relation nominals available and most of the time we not only need to look at the nominal but also need to refer to the contexts to judge. Sampling evaluation normally picks the top ranked outputs or randomly picks some of the outputs to estimate the learning quality. However, we learned hundreds of nominals and thousands of contexts and we believe sampling is the not the best way to reflect the overall learning quality of our system.

We then decide to incorporate the learned lists of nominals and contexts as features into a SVM-based relation extraction system to see if its performance can be boosted or not.

[3] The reason for using N_{+1} instead of the count of x in the corpus in computing *P(x)* is that we do not count bigrams which cross sentence boundaries. Please refer to Inkpen and Hirst (2002) for a detailed description of all these statistics.

[4] Please refer to Grishman et al. (2005) and http://cs.nyu.edu/grishman/jet/license.html

4. Using Relation Nominals to Improve Supervised Relation Extraction

We first build a SVM-based relation extraction system as our baseline system trained on the annotated data from the 2004 ACE (Automatic Content Extraction) evaluation[5], which is commonly used by researchers to report and compare system performance. We then extract three types of features (to be explained in section 4.3) from nominal lists learned by two-stage bootstrapping and incorporate them into the baseline.

4.1 ACE Terminology

In ACE vocabulary, entities are objects and mentions are references to them. Entities can be of 5 types: person, organization, location, facility, and geo-political entity. Mentions have levels: name, pronoun or nominal.

The ACE Relation Detection and Characterization (RDC) task detects relations between entities. As in Example 1, there is an EMP-ORG relation between *analyst* and *the Council on Foreign Relations*, where *analyst* is a mention and referenced to the entity *Lawrence Korb*.

Example 1: *Lawrence Korb, an analyst at the Council on Foreign Relations who was assistant defense secretary under former President Ronald Reagan.*

4.2 Baseline System

Our baseline system is a duplicated system of (Zhou et al., 2005). We adopt most of their features except the *personal relative trigger word list* since it is not relevant to our EMP-ORG scenario.

Features can be characterized into 5 categories: lexical, base phrase chunking, dependency tree, parse tree and semantic resources such as country name list. Each category contains several subtypes of features. There are 41 subtypes of features in our system (the only two subtypes features we are not adapting are extracted from the *personal relative trigger word list*). Please refer to (Zhou et al., 2005) for a detailed description of features.

It's important to point out that we train our system on relation type EMP-ORG not its subtypes. We decode positive and negative instances in the following way: if 2 mentions have an EMP-ORG relation annotation in the ACE key file, we then build a positive instance; any 2 mentions within a sentence which do not have EMP-ORG relation will be built as a negative instance.

We use the official ACE 2004 training and evaluation data from LDC for experiment. We exclude the 8 fisher transcript files from training data. So we use in total 635 files and there are 2,874/773,412 positive/negative instances.

We use the SVM-light[6] package as our machine learning method. We use a linear kernel and do 10-fold cross-validation in our baseline and all the other experiments.

4.3 Features Extracted from Nominal Lists

We extract three types of features from our two-stage bootstrapping learned nominal lists.

4.3.1 InNomList

This is a binary feature which checks whether the *head* of *mention 1* is in our learned list or not. We combine it with the entity types of both mentions to prevent the feature from being too general. In Example 1, the *head* of *mention 1* is *analyst* and it is in our learned nominal list. So for Example 1, we construct the following feature:

InNomList=PERSON-ORG-true

4.3.2 ContextNomList

For each learned nominal in our lists, we assume it is the *head* of *mention 1* in a learned pattern. We then generate a list *L* of words between this nominal and the *head* of *mention 2* which is an ORG in our pattern. For example, *L = {at, of, for}* for nominal *analyst* and its associated patterns { *PERSON, an *NN* at ORG; PERSON , chief *NN* of ORG*; *PERSON , managing *NN* at ORG*; *PERSON , chief financial *NN* for ORG* }

Given a positive/negative instance, we first extract the words between the heads of both mentions; Then we check if the words are in *L* or not. In Example 1, *at* is the word between *analyst* and *the Council* and is in *L*, so we construct the following feature:

ContextNomList=PERSON--ORG--at (combined with entity types)

4.3.3 ModNomList

For each learned nominal, we generate a list *L* of its modifiers from its associated patterns. *L = { an, chief, managing, financial }* for the *analyst* example. Given an instance, we extract the words before the head of *mention 1* and check if they are in our modifier list or not. We generate the following feature for Example 1:

ModNomList=an-PERSON (combined with entity type of mention 1)

4.4 Experiments and Results

We conducted 4 experiments. In the rest of this paper, we will refer to the system which added to *Baseline* features from ***BMI*** learned nominal list as *System BMI*, the system which added features from ***Dice*** learned list as *System Dice* and the system which added features from the merged list of ***BMI*** and ***Dice*** as *System Combined*.

[5] http://www.itl.nist.gov/iad/mig//tests/ace/

[6] http://svmlight.joachims.org/

Figure 2, 3 and 4 show Precision, Recall and F-measure of our experiments, where 1/2/3/4 mean *System Baseline/BMI/Dice/Combined.*

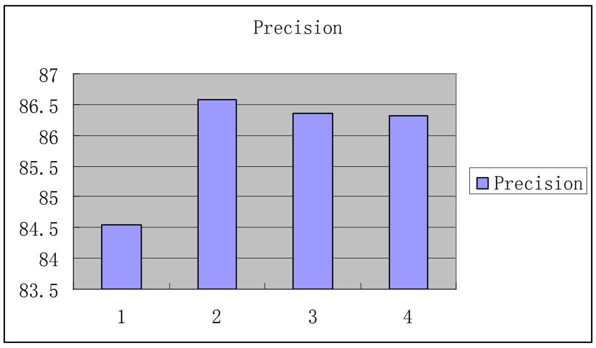

Figure 2. Precison

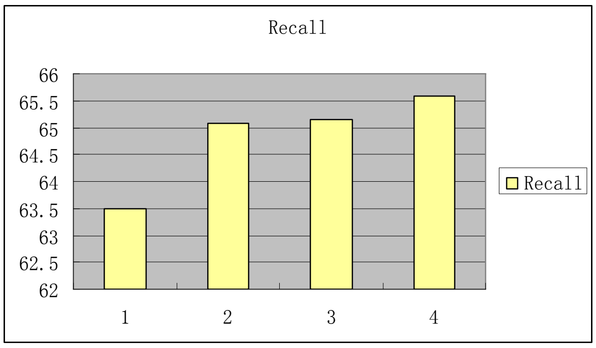

Figure 3. Recall

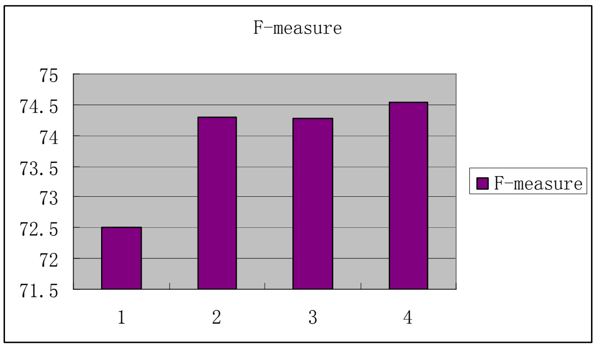

Figure 4. F-measure

We first notice that our baseline achieves similar results as reported in (Yong and Su 2008) whose baseline is also a duplicated system of (Zhou et al., 2005)[7]. Though, we should point out that we achieve a slightly higher precision and a lower recall than (Yong and Su 2008). This is first because we use different experiment settings. Also, it is probably caused by different tools used for generating features. We use the same Perl script[8] used in (Zhou et al., 2005) to derive base phrase chunk information from full parse tree. But we use Jet for tokenization and the Charniak parser for full parsing.

The results show that:

- Both *MI* and *Dice* improves the Baseline while *MI* achieves slightly higher precision than *Dice* and *Dice* achieves slightly better recall than *MI*. When we check our nominal list, we found that *MI* is more cautious than *Dice* in expanding patterns for a nominal. Table 3 shows the top 10 ranked nominals and the number of associated patterns for both *MI* and *Dice*. It may be that *Dice* achieves better recall in part because there are more patterns being used for generating features. However, *Dice* would sacrifice precision a little bit because it might also include some bad patterns and thus some bad nominals which are generated by these bad patterns.
- The merged list gives the best recall and F-measure. There are 200 nominals from the merged list which are used during feature decoding while there are 152/156 nominals from *MI/Dice* being used. We can also imagine that the merged list uses more patterns for feature decoding.

Table 3. Top 10 ranked nominals and number of associated patterns

Nominal (*MI*)	Number of patterns (*MI*)	Nominal (*Dice*)	Number of patterns (*Dice*)
executive	109	executive	126
scientist	32	economist	66
economist	28	analyst	58
editor	24	editor	46
architect	24	officer	43
minister	23	scientist	40
justice	20	engineer	37
counsel	20	architect	36
judge	20	investigator	35
designer	20	counsel	34
Total:	320		521

[7] Zhou et al. (2005) reported result for ACE 2003 data, not ACE 2004 data.

[8] http://ilk.uvt.nl/team/sabine/chunklink/README.html

- Although there are some differences between the MI list and Dice list, they give similar improvements and the combined list gives even more improvements. This suggests that MI and Dice lists are complementary to each other and both of them are reliable.

5. Conclusions and Future Work

Using bootstrapping to extract relations which are normally general and contain subtypes of relations is challenging. This paper proposes a two-stage bootstrapping learning algorithm for addressing this problem. We show a case study in EMP-ORG and observe 2% F-measure improvement when we incorporate features extracted from two-stage bootstrapping learned nominals and context into a supervised relation extraction system.

Our immediate future work involves testing this method on more relation types. Our current system relies on tokens between name pairs. Incorporating parsing information into two-stage bootstrapping would be challenging yet interesting future work.

6. Acknowledgements

This research was supported by the National Science Foundation under Grant IIS-00325657. This paper does not necessarily reflect the position of the U.S. Government. We would like to thank Prof. Ralph Grishman for his research guidance. We would also like to thank Prof. Satoshi Sekine, Prof. Adam Meyers and Prof. Heng Ji for their useful suggestions.

7. References

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain text collections. In Proceedings of the 5th ACM International Conference on Digital Libraries, 2000.

[2] S. Brin. Extracting patterns and relations from the World-Wide Web. In Proceedings of the 1998 International Workshop on the Web and Databases (WebDB'98), March 1998.

[3] R. Grishman, D. Westbrook and A. Meyers. 2005. NYU's English ACE 2005 System Description. ACE 2005 PI Workshop. Washington, US.

[4] DZ. Inkpen and G. Hirst. 2002. Acquiring collocations for lexical choice between near synonyms. In Unsupervised Lexical Acquisition: Proceedings of the Workshop of the ACL Special Interest Group on the Lexicon (SIGLEX), pp. 67–76, Philadelphia, Pennsylvania.

[5] N. Kambhatla. 2004. Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics.

[6] E. Riloff and R. Jones. 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99).

[7] S. Sekine. 2008. A Linguistic Knowledge Discovery Tool: Very Large Ngram Database Search with Arbitrary Wildcards. In proceedings of the 22nd International Conference on Computational Linguistics, Manchester, England.

[8] SWK Yong and J. Su. An Effective Method of Using Web Based Information for Relation Extraction. In proceedings of 3rd International Joint Conference of Natural Language Processing (IJCNLP2008), P350-357, Hyderabad, India.

[9] S. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. In Proceedings of ACL.

[10] G. Zhou, J. Su, J. Zhang and M. Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In proceedings of 43th Annual Meeting of the Association for Computational Linguistics. USA.