Clustering for Unsupervised Relation Identification

Benjamin Rosenfeld
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
grurgrur@gmail.com

Ronen Feldman
Information Systems
HU School of Business,
Hebrew University, Jerusalem, Israel
ronen.feldman@huji.ac.il

ABSTRACT
Unsupervised Relation Identification is the task of automatically discovering interesting relations between entities in a large text corpora. Relations are identified by clustering the frequently co-occurring pairs of entities in such a way that pairs occurring in similar contexts end up belonging to the same clusters. In this paper we compare several clustering setups, some of them novel and others already tried. The setups include feature extraction and selection methods and clustering algorithms. In order to do the comparison, we develop a clustering evaluation metric, specifically adapted for the relation identification task. Our experiments demonstrate significant superiority of the single-linkage hierarchical clustering with the novel threshold selection technique over the other tested clustering algorithms. Also, the experiments indicate that for successful relation identification it is important to use rich complex features of two kinds: features that test both relation slots together ("relation features"), and features that test only one slot each ("entity features"). We have found that using both kinds of features with the best of the algorithms produces very high-precision results, significantly improving over the previous work.

Categories and Subject Descriptors
I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.6 [**Artificial Intelligence**]: Learning

General Terms
Algorithms, Measurement, Performance, Experimentation.

Keywords
Unsupervised Relation Identification, Clustering, Relation Learning, Information Extraction.

1. INTRODUCTION
Information Extraction (IE) is the task of extracting factual assertions from text. Most IE systems rely on knowledge engineering or on machine learning to generate the "task model", which is subsequently used for extracting instances of entities and relations from new text. In the knowledge engineering approach the model (usually in the form of extraction rules) is created manually, and in the machine learning approach the model is learned automatically from a manually labeled training text. Both approaches require substantial human effort, particularly when applied to the broad range of documents, entities, and relations existing in the Web. In order to minimize the manual effort necessary to build IE systems, semi-supervised systems were developed, able to learn the task model using only a large unlabeled corpus and a small set of *seeds* – known true instances of the target entity or relation.

Even semi-supervision can be costly for relation extraction at Web scale, because of a large number of different domains and possible relation types. *Prevemptive Information Extraction* approach was introduced in [10] as a remedy to this problem. Under this approach, there is a preparation stage, during which the IE system automatically discovers all relations, extracts their instances, and puts them into tables, which can be mined later, in response to the user queries or for other purposes.

This *preemptive IE* approach relies on *Unsupervised Relation Identification* (URI) – an automatic discovery of all interesting relations in a large body of text. Any relation can be identified by a representative subset of its instances. Thus, the task of URI is to provide such representative set of instances for each of the relations it discovers.

URI systems usually work in the following way: First, the set of candidate relationships is generated. It may simply consist of all pairs of entities that frequently co-occur in same sentences. Then, the system analyzes the sentences in which the candidates occur, and produces a measure of similarity between the candidates, based on similarity of the contexts in which they appear. Then, the candidates are clustered using some general-purpose clustering algorithm. Finally, the clusters are pruned, and poor clusters are discarded.

In [8] we had shown that the clusters discovered by URI can be successfully used for seeding a semi-supervised relation extraction system, producing performance comparable to the performance of human-selected seeds. Thus, it makes sense to separate the tasks of relation identification and actual relation extraction. The relation identification stage should find frequent high-precision seeds for the relations, while the extraction stage should extract many more relation instances, gaining recall and balancing it against required precision.

In this paper we focus on the URI task. We compare several clustering algorithms, and several feature extraction and selection methods, evaluating them using a novel evaluation metric, which

CIKM'07, November 6-8, 2007, Lisboa, Portugal.

is adapted for evaluating precision-oriented URI clustering results. Our findings demonstrate:

- Effectiveness of the single-linkage HAC (Hierarchical Agglomerative Clustering) algorithm with our novel threshold-selection technique, and its superiority over the other common clustering algorithms – average-linkage HAC, complete-linkage HAC, and *K*-Means – even when using the best possible thresholds for the HAC-s, and the best number of clusters for the *K*-Means.
- Importance of the complexity of the context features that are used for representing the candidates. We show that only sufficiently powerful context patterns produce good results.
- Importance of using both the features that test individual entities in the candidate pairs (entity features), and the features that relate the entities inside pairs to each other (relation features).
- Effectiveness of the separability-based feature selection method, which allows to reduce the dimensionality of the feature space by two orders of magnitude, with little or no deterioration of clustering quality.

In our experiments we consistently avoid using supervised named entity recognition (NER) systems. Good NER components only exist for common and very general entity types, such as *Person*, *Organization*, and *Location*. For some relations, the types of attributes are less common, and no ready NER components (or labeled training sets) are available. Also, some Web relation extraction systems (e.g., KnowItAll [3]) do not use supervised NER components even for known entity types, because such components are usually domain-specific and may perform poorly on cross-domain text collections extracted from the Web.

Consequently, the entities in our candidate pairs are simply proper noun phrase heads, extracted by a general-purpose noun phrase chunker (trained on a dataset from the CoNLL-2000 shared task [11]), with the entity boundaries fixed by corpus statistics-based methods ([9]).

Nevertheless, we show that the clusterings produced by the best of the strategies approach perfect precision, sufficient for seeding the semi-supervised relation extraction systems.

The rest of the paper is structured as follows: in the next section we describe the related work. Then we present the details of the architecture of our URI system, and its various components. Then we describe our experiments, discuss their results, and conclude.

2. RELATED WORK

The field of URI is very recent, and there is only a small number of works available.

Discovering relations by clustering pairs of co-occuring entities represented as vectors of context features was introduced by Hasegawa, Sekine et al. [6]. They used a complete-linkage hierarchical clustering algorithm, with a very simple representation of contexts – the features were the words that appear in sentences between the entities of the candidate pairs. Consequently, their results had a relatively low precision and were unsuitable for bootstrapping, which usually amplifies any noise in the set of seeds.

Chen, Ji et al. [1] used the same context representation, but added an entropy-based feature ranking and selection step. They also used the *K*-Means algorithm instead of HAC, introducing a stability-based criterion for estimating the number of clusters.

Shinyama and Sekine [10] used simple predicate-argument patterns around the entities of candidate pairs. Their system works on news articles, and improves its accuracy by looking at multiple news sources describing the same event.

The authors in [8] first attempted to use the results of URI for seeding a semi-supervised relation extraction system. Using rich surface patterns for representing the context features we achieved a limited success in discovering and subsequently extracting a small number of relations in the financial news domain.

In this paper we build upon our previous work, experimenting with different clustering algorithms and representations, comparing them to each other using a novel evaluation metric, and producing an URI system that substantially outperforms all of the mentioned systems.

3. URI FRAMEWORK

In this work we are primarily concerned with the part of a URI system that does clustering, and we do not deal with candidate selection, nor with complex post-processing such as generation of names for the discovered relations.

Thus, we can assume that the input to the system is a set of entities E and a set of relation candidates (pairs of entities) $C \subset E \times E$. Also, there is a set of *contexts* S_e for each entity $e \in E$. Each context of an entity is a sentence in which the entity appears at least once. The set of contexts S_c of a candidate $c = (e, e') \in C$ consists of sentences in which both entities of the candidate appear: $S_c = S_e \cap S_{e'}$.

The output of the system is a partitioning of the candidates into disjoint clusters $C = C_1 \cup C_2 \ldots \cup C_n \cup G$, where the clusters C_i identify the discovered relations, and the special cluster G (for "garbage") gathers all unclassified candidates. The garbage cluster is an essential element of the scheme, since usually many of the candidate pairs do not belong to any well-defined general relation.

The URI process consists of three main stages: (1) feature extraction – representing the candidates as vectors in some feature space, (2) clustering of vectors, and (3) pruning of the resulting clusters. These stages and the various components performing them are described below.

3.1 Feature Extraction

The goal of this stage is to prepare the information necessary for the clustering algorithms: a representation of the candidates as vectors in some feature space, with a suitable measure of distance between vectors.

The representation and the distance (or similarity measure) must be chosen in such a way that under this measure the candidates that often appear in similar contexts would be close to each other, and conversely, the candidates that rarely appear in similar contexts would be far apart. Consequently, we base the

representation directly on features of the contexts, which can abstractly be defined as follows:

Let $e \in E$ be an entity, and $s \in S_e$ be one of its contexts. We *m-slot-mark* the entity e in s by substituting a special token "<*m*>" for one of the instances of e inside s. Similarly, we *slot-mark* the candidate pair $c = (e, e')$ inside its context $s \in S_c$ by substituting the token "<1>" for one of the instances of e and the token "<2>" for one of the instances of e' inside s. We let $MS(e, m)$ denote the set of all m-slot-marked contexts of the entity e, and $MS(c)$ the set of all slot-marked contexts of the candidate c. Finally, for $c = (e, e')$ we let

$$M(c) = MS(e,1) \cup MS(e',2) \cup MS(c)$$

denote the set of all slot-marked contexts related to c, and

$$M = \bigcup_{c \in C} M(c)$$

the set of all slot-marked contexts of all candidates.

A *context feature* is a function $f : M \to \{0, 1\}$. Each such function can be thought of as checking for presence or absence of some feature in the contexts of entities and pairs. A family of such functions defines some measure of *similarity* between contexts – two contexts are more similar when the values of more functions coincide on them. We extend this similarity to similarity between candidates by defining their representation as follows:

Let F be a set of context features. Then, the candidate c is represented in the *feature space* $\boldsymbol{R}^{|F|}$, indexed by F, by a vector $\boldsymbol{v}(c)$ with components

$$v_f(c) = \sum_{s \in M(c)} f(s)$$

so the value of the f-th component of the representation of c equals to the number of times the context feature f appears among the contexts of c.

For similarity between two vectors in the feature space we use the common cosine measure

$$Sim(\mathbf{v}_1, \mathbf{v}_2) = \cos(\mathbf{v}_1, \mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|}$$

which ignores the relative lengths of vectors.

Naturally, the utility of such representation depends on the quality of the context features. For our features we use *surface patterns*, similar to the ones we used in [8]. Each such pattern is a sequence of tokens (including the special slot-mark tokens), and *skips*, which indicate gaps in the pattern. Given a pattern p, we define the context feature f_p corresponding to the pattern by

$$f_p(s) = \begin{cases} 1, & \text{if } p \text{ matches } s, \\ 0, & \text{otherwise.} \end{cases}$$

A pattern p matches the context s iff all of the tokens of p appear in s in the correct order. The tokens of p must also be adjacent to each other inside s, except where gaps in the pattern are allowed (by *skips* in p), and where arbitrary sequences of tokens may appear between the tokens of p.

As an example to the concepts defined above, consider the sentence

> *IBM, based in Armonk, New York, said it would finance the acquisition of Lotus.*

This sentence would belong to the context sets S_e of the entities e = *"IBM"*, *"Armonk"*, *"New York"*, and *"Lotus"*, and to the context sets of all candidate pairs built from these entities. If we 1-slot-mark the entity *"IBM"* inside the sentence, we get

> *<1> , based in Armonk , New York , said it would finance the acquisition of Lotus .*

which would belong to the set $MS($*"IBM"*$, 1)$ and to the sets $M(c)$ for all candidate pairs c that have *"IBM"* as their first argument. The context would be matched by the pattern

> *<1> , based in*

and would not be matched by the pattern

> *<1> said*

If we further 2-slot-mark the entity *"New York"*, we get a slot-marked context

> *<1> , based in Armonk , <2> , said it would finance the acquisition of Lotus .*

which would belong to the sets $MS(c)$ and $M(c)$, where c is the pair (*"IBM"*, *"New York"*). Then, the pattern

> *<1> , based in <2>*

would not match this slot-marked context, whereas the pattern

> *<1> , based in * <2>*

where the asterisk denotes a skip, would.

We generate the set of surface patterns by a suitable modification of the Apriori association mining algorithm. First we extract all sequences of tokens (without gaps) that appear in the set M of all contexts with frequency greater than a given minimal support value. Then we mine the contexts (as ordered sets of such sequences) for frequent itemsets. The result is the set of all surface patterns sufficiently supported by the set of contexts.

Despite their simplicity, the surface patterns are sufficiently general to capture many recurring properties of contexts, and are sufficient to produce good-quality clusterings, as our experiments demonstrate. They are also more general than all the context features used in the works mentioned in the Related Work section, and this allows us to simulate their performance by suitably restricting the form of our patterns. For example, the bag-of-words feature space of [6] and [1] can be simulated using patterns of the form

> *<m> * w * <m'>*,

while the predicate argument patterns of [10] can be simulated by the surface patterns of the form

> $w_1\ w_2 \ldots w_i$ *<m>* $w_{i+1} \ldots w_k$.

In [8], we used surface patterns of unrestricted form. However, we used a different method of pattern extraction – pairwise context generalization. The main drawback of that method is its computational complexity. Also, we did not use the contexts of

individual entities. Thus, the feature space of [8] can be simulated in the current system by using the patterns of the form

$$\ldots \langle m \rangle \ldots \langle m' \rangle \ldots$$

3.2 Feature Selection

The number of surface patterns generated at the feature extraction step can be very large. In order to reduce the computation costs of the clustering stage, it is beneficial to remove the useless features. We cannot use the common classification feature ranking measures such as Information Gain or χ^2 for the clustering task, since these measures rely on knowledge of the classification of the training set. However, following [1], we can use a feature ranking based on the observation of [2] that good clustering features should improve the separability of the dataset – make points that are close together still closer, and points that are far from each other still farther apart.

Using this intuition, we define the score of a feature *f* by

$$Score(f) = E - E_{-f}$$

where

$$E = -\sum_{c_i, c_j \in C} S_{ij} \log S_{ij} + (1 - S_{ij}) \log(1 - S_{ij})$$

$$E_{-f} = -\sum_{c_i, c_j \in C} S_{ij}^{f} \log S_{ij}^{f} + (1 - S_{ij}^{f}) \log(1 - S_{ij}^{f})$$

and where S_{ij} is the similarity between c_i and c_j in the full feature space, normalized to [0..1], and S_{ij}^{f} is the similarity between c_i and c_j after removing the feature *f*, also normalized to [0..1].

Given a ranking of features, we can select *N* top-scoring ones and remove the rest. In the experimental section we demonstrate that it is sometimes possible to reduce the number of features by a factor of 100 without losing the quality of clustering.

As defined, the merit of the feature selection scheme is questionable, since its computational complexity can be greater than the complexity of actual clustering. However, Dash and Liu argue in [2] that separability-based scores of features can be approximated well by calculating them over small samples of the whole dataset. We do not test this hypothesis in this work, leaving it for further research.

3.3 Clustering

We test and compare two common general-purpose clustering algorithms in our experiments: the hierarchical agglomerative clustering (HAC) algorithm, and the partitioning *K*-Means algorithm.

The HAC algorithm starts with all datapoints in separate clusters, and proceeds to iteratively merge the most similar clusters, until all pairwise distances between the clusters become greater than a prespecified threshold. There are three common ways of calculating the distance (or similarity) between clusters in terms of pairwise distances between their datapoints: *single linkage*, *average linkage* and *complete linkage*. Using these three linkages, the similarity between two clusters is, respectively, the *maximum*, the *average*, and the *minimum* of the similarities between their datapoints.

The *K*-Means algorithm starts with a set of *k* seed datapoints that define the initial clusters. At each iteration until convergence, the algorithm calculates the centroid of each one of the *k* clusters, and reassigns every datapoint to the cluster that has the closest centroid.

Both the HAC and the *K*-Means algorithms have parameters that significantly affect the quality of clustering. HAC has the linkage type parameter and the similarity threshold parameter, which specifies when to stop merging the clusters. In all previous works, the HAC was always used with a complete linkage, and with a small fixed threshold.

K-Means, on the other hand, depends on the choice of the seeds and on the number *k* of clusters, which must be known in advance. Usually, the problem of seeds dependence is solved by running the algorithm several times with different random choice of seeds, and then selecting the best clustering among the results, according to some internal consistency measure. As for estimating the number of clusters, Chen, et al [1] advocate using stability-based criterions.

In our experiments we surprisingly found that the single linkage HAC not only outperforms both the HAC with other linkage types and the *K*-Means, but also has a natural stopping criterion: "stop merging the clusters when the average similarity between the datapoints of the clusters being merged becomes smaller than α times the maximal similarity between them", where $\alpha < 1$ is a constant, for which we use 1/2.

Although this stopping criterion does not always find the best clustering possible for the single linkage HAC, it usually finds a clustering within a small margin of the best. And in almost all cases, the clustering it generates is better than the ones produced by the HAC with other linkage types, or by the *K*-Means, even with the best choice of their parameters. (See the experimental section for the details).

3.4 Pruning the Clusters

In this paper we are primarily concerned with the clustering task. Thus, we utilize a simple and crude method of pruning: we disband all clusters that contain less than a given number of candidates, 4 in our experiments. The released candidates are put into the garbage cluster.

This method of pruning makes sense for the URI task, which after all attempts to discover relatively common and open-ended relations.

4. EXPERIMENTS

We performed a set of experiments to compare the feature extraction and selection methods and the clustering algorithms described above. We shall first describe our evaluation criterion, and then proceed to the actual experiments.

4.1 Evaluation

There are various methods for evaluating clustering results [5]. For the URI task we have to use an *external* criterion, since we would like the clusters to agree with human intuition as to what

constitutes an interesting relation. Consequently, in order to score a given clustering X of a set of candidates, we first manually cluster the candidates into a *model clustering* MX. Then we calculate the square root of the *Jackard coefficient*

$JC(\mathrm{X}, \mathrm{MX}) = (SS / (SS + SD + DS))^{1/2}$,

where

- SS = the number of pairs of candidates that co-occur (appear together in some cluster) in both X and MX,
- SD = the number of pairs of candidates that co-occur in X but not in MX,
- DS = the number of pairs of candidates that co-occur in MX but not in X.

The closer X is to MX, the higher is $JC(\mathrm{X}, \mathrm{MX})$, which can therefore be used as a score for comparing different clusterings.

However, the URI task is different from other clustering tasks, because of the existence of a large number of unclassified datapoints, gathered into a single specially designated garbage cluster. The garbage cluster contains the entity pairs that do not belong to any recognizable relation, which happens either if the co-occurrence of the entities is accidental, or if the relation between them is too idiosyncratic.

Since the URI task places much higher value on precision than on recall, the garbage cluster must be evaluated separately from the other clusters, which represent actual relations. In particular, it should be less costly to put a candidate belonging to a relation into the garbage cluster, than to allow a wrong candidate to get in.

Thus, the garbage cluster is treated differently in two ways: first, the definition of "co-occurrence" does not include the garbage cluster – the candidates inside the garbage cluster are considered "not co-occuring". And second, pairs of candidates that co-occur in MX, but at least one one of which is in the "garbage" in X, are counted in SD not as "1"-s, but as smaller constants α. We experimentally found that the precise value of α does not matter for qualitative ordering of performance of various clustering setups, as long as the value is noticeably smaller than one. In our experiments we use $\alpha = 0.3$.

4.2 Experimental Setup

For the experiments we used three different text corpora. One of them is the corpus that was used for evaluating the semi-supervised SRES [4] and KnowItAll [3] systems. The corpus consists of a concatenation of four separate corpora, for the *Acquisition*, *Merger*, *Mayor_Of*, and *CEO_Of* relations. Each sentence in the corpus contains at least one keyword related to one of the four relations. Although the four corpora were processed separately by the semi-supervised systems, we combined them together for evaluating URI, in order to see how well the system is able to find the relations and to distinguish between them. This corpus is denoted ACMM.

The second corpus is a subset of the RCV1 corpus of Reuters news [7], which included news articles on various economics-related topics.

The third corpus is a one year (1995) of The New York Times, which was also used in the original work of Hasegawa, Sekine et al. [6].

The corpora were processed by a general-purpose noun phrase chunker, trained on the labeled data for the CoNLL-2000 shared task [11]. All pairs of proper noun phrases that co-occurred sufficiently frequently (30 times or more) were processed by a corpus statistics-based boundary fixing component (described in [9]) in order to reduce the noise and the amount of garbage, and the results were placed into the *initial sets* of candidates. This process produced several thousands of candidate pairs for each of the corpus, which is currently beyond the computing capabilities of our system, due to space problems appearing because of a huge number of features. Therefore, we randomly selected a thousand of candidates from each of the initial sets, and put them into the *working candidate sets*, on which the actual clustering experiments were performed.

Since it is too costly to manually cluster even one thousand candidates, we further randomly selected a subset of 200 candidates from each of the working sets, and manually clustered these subsets. During the experiments, the clustering algorithms were run over the full working sets of candidates, but only the selected 200 candidates were used for scoring the results.

4.3 Experiment 1: Comparing the algorithms and the feature sets

In the first series of experiments we compared the performance of various clustering algorithms and various feature sets described above. The results are shown in the Table 1. And in the Table 2 we list the number of features the system extracted in each of the setups.

The K-Means results are the best scores from a large number of runs – five runs for each different k, where in each run the first seed was selected randomly, and the subsequent k-1 seeds were selected by maximizing the distances between them. (Random selection of all seeds produced worse results).

All HAC results, except for the "HAC single*", are the best-scoring results among all possible settings of the threshold parameter.

The "HAC single*" denotes the HAC with the single linkage and with the "Average/Max similarity < 1/2" stopping criterion.

Several conclusions can be drawn from the results. First, we can see that the extent and the generality of features are very important. The bag-of-words representation is clearly poor and insufficient. Both "Argument-only" (context features related to one of the entities in a pair, as in [10]) and "Combination only" (context features related to both entities together, as in [8]) perform reasonably well, but still worse than the full-featured setting.

Regarding the algorithms, the HAC with single linkage outperforms both K-Means and the other variants of HAC in almost all cases. Also, we can see that the "Average/Max similarity < 1/2" stopping criterion, which is extremely simple and needs no additional computation, performs surprisingly well, always producing results within a very small margin of the best possible.

Table 1: Performance (JC-score) of several different feature sets and clustering algorithms.

Features	Algorithm	ACMM	RCV1	NYT95
Bag of words	K-Means	0.626	**0.489**	**0.509**
	HAC single	**0.669**	0.376	0.442
	HAC single*	**0.658**	0.341	0.394
	HAC ave	0.644	0.459	0.447
	HAC comp	0.648	0.418	0.375
Arguments Only	K-Means	0.751	0.808	0.798
	HAC single	**0.775**	0.819	**0.866**
	HAC single*	**0.769**	0.807	**0.866**
	HAC ave	0.762	**0.841**	0.822
	HAC comp	0.737	0.788	0.781

Features	Algorithm	ACMM	RCV1	NYT95
Combination Only	K-Means	0.725	0.660	0.691
	HAC single	**0.805**	**0.878**	**0.871**
	HAC single*	**0.786**	**0.873**	**0.843**
	HAC ave	0.765	0.796	0.823
	HAC comp	0.702	0.648	0.648
All Features	K-Means	0.793	0.848	0.792
	HAC single	**0.813**	**0.875**	**0.934**
	HAC single*	**0.812**	**0.850**	**0.921**
	HAC ave	0.803	0.849	0.829
	HAC comp	0.766	0.804	0.771

Table 2: The number of features for different feature extraction methods.

Features	ACMM	RCV1	NYT95
Bag-of-words	1433	2664	1869
Arguments only	40307	52252	25418
Pair only	100408	122191	36806
All features	140715	174443	62224

Table 3: Feature selection performance

Corpus	Features Count	HAC Single* Performance			
		All	Top 5000	Top 2000	Top 1000
ACMM	142846	0.812	0.813	0.794	0.784
R2	175830	0.850	0.832	0.832	0.829
NYT95	131974	0.921	0.944	0.903	0.897

4.4 Experiment 2: Feature Selection

In this series of experiments we test the separability-based feature selection scheme. The results are summarized in the Table 3. We show only the results for the HAC with single-linkage and AS/MS stopping criterion, as this is the best-performing complete algorithm:

As can be seen, the separability-based feature selection is very effective, reducing the dimensionality of the feature space by a factor of 20 with no deterioration in results, and by a factor of 100 with only a very slight deterioration.

4.5 Identified Relations

In the Table 4 we list cluster-by-cluster the relations identified by the system in the New York Times 1995 corpus, together with their sizes and the number of mistakes.

As can be seen, the system is very precise, introducing almost no wrong candidates into the relations. The two mistakes, each by 4 wrong candidates, come from incorrect merging of correct clusters.

The system also produced several "closed" clusters, which list the same pair of candidates in different wordings and/or in a different order. For example, the cluster *FinMinistry-Japan* contains pairs (*"Finance Ministry", "Japan"*), and (*"Ministry of Finance", "Japan"*). The system of course has no knowledge that the entities *"Finance Ministry"* and *"Ministry of Finance"* are actually the same.

Finally, there are four "garbage" clusters of various sizes, which include pairs unrelated to each other and pairs that do not belong to any well-defined relation.

However, we believe that the closed clusters and the garbage clusters do not significantly harm the intended use of the system,

Table 4: Cluster-by-cluster decription of relations identified in the NYT95 corpus.

Cluster Name	Correct	Wrong	Cluster Name	Correct	Wrong	Cluster Name	Correct	Wrong
RelatedCountries	43	0	PresidentCountry	4	0	**Closed clusters**		
CityOfState	41	0	ChairmanOfOrg	4	0	Fed-OpenMarket	3	0
	14	4	UniversityIn	4	0	Deutsch-France Telekom	2	0
SenatorOfState	16	0	NominantsGOP	2	0	Ford-Chrysler	2	0
	6	0		2	0	Chase-Chemical	2	0
EmployedIn	28	0	StateGovernor	3	0	Serbia-Croatia	2	0
	16	0	CountryWTO	3	0	GOP-America	2	0
OrgLocation	6	0	Israel-Territories	3	0	Karadzic-Mladic	2	0
	2	0	PrimeMinister	3	0	Sinn Fein-IRA	2	0
CityOfCompany	15	4	OrgComissioner	3	0	FinMinistry-Japan	2	0
OrgChairman	12	0	Acquisition	3	0	**Bad clusters**		
TeamCoach	10	0	ParentCompanyOf	3	0	garbage		6
NewsNetworks	6	0	CapitalPresident	2	0	garbage		5
SenateLeaderOf	4	0	SameTeamPlayers	2	0	garbage		2
	2	0	TeamOwner	2	0	garbage		2
CountryPresident	5	0	FBI-OfficeIn	2	0			
AnalystIn	5	0	SchoolProfessor	2	0			
Team-Player	4	0	OrgAlias	2	0	**Total**	307	23
State-Company	4	0				**Precision**	0.930	
						Precision excluding bad and closed clusters	0.973	

Table 5: Cluster details: two candidate pairs for each cluster, and an example context for each pair.

ClusterName	Pair	Example context
RelatedCountries	Argentina - Brazil	Once a hostile rival of <1> , <2> has become its largest...
	Australia - New Zealand	… considering a tour of <1> and <2> in the fall.
CityOfState	Alexandria - Va.	A grand jury in <1> , <2> reportedly has been impaneled…
	Arlington - Texas	"We ' re looking at <1> , <2> ," added…
SenatorOfState	Bill Archer - Texas	Republican Reps. <1> of <2> , chairman of the…
	John Kasich - Ohio	Budget Committee chairman <1> , R- <2> , reminded…
EmployedIn	Andrew Brenner - Nomura	...said <1> , senior trader at <2> .
	Belal Khan - Bank of Tokyo	...said <1> , chief customer dealer at <2> in New York .
CityOfCompany	Armonk - IBM	...shares of <1> , New York -based <2> rallied...
	Boston - Fidelity Investments	Protest from <1> -based <2> was crucial to…
OrgChairman	Boeing - Frank Shrontz	As recently as December , <1> Chairman <2> predicted that…
	Chrysler - Robert Eaton	<1> Chairman <2> said last month…
TeamCoach	Arkansas - Nolan Richardson	<1> basketball coach <2> , trying to explain why his…
	Cowboys - Barry Switzer	However , <1> coach <2> said there 's no way for anyone…

which is identification of relations for further extraction by semi-supervised systems. If these clusters are excluded, the precision of the system is nearly perfect.

In Table 5 we show further details for several top clusters: we list two candidate pairs (alphabetically first) placed into the cluster, and a single context (chronologically first) extracted for each listed candidate. As can be seen from this table, the generated clusters are semantically reasonable.

5. CONCLUSIONS AND FUTURE RESEARCH

We have presented the results of several experiments, comparing different clustering algorithms and different feature extraction and selection methods for unsupervised relation identification, using an evaluation metric adapted for the task.

The experiments demonstrate superior performance of the single linkage HAC clustering algorithm with the average/maximal similarity stopping criterion.

We also demonstrated the importance of using complex features, and relying on features that are based both on individual entities and on combination of entities within pairs.

Unlike most of the previous works, our system performs without a separate named-entity recognition component, using only a general-purpose noun phrase chunker. Thus, all of the entities initially belong to the same large set. Nevertheless, there are almost no argument type mistakes in the final clusters. Also, although the general-purpose noun phrase chunker makes many mistakes, candidate pairs containing bad entities end up placed into the garbage cluster.

In the future research we plan to test the sample-based feature selection, which if successful would allow us to use much bigger initial sets of features, and thus to work with bigger candidate sets. Currently, very big candidate sets are inaccessible to our system, because the number of features that need to be extracted is too large.

We also plan to test our clustering scheme in other unsupervised clustering settings, such as clustering of relations between individual words, for the field of unsupervised language acquisition.

6. ACKNOWLEDGEMENTS

Some of the data sets were provided by the KnowItAll project at the University of Washington's Turing Center. We thank Oren Etzioni and Stephen Soderland for helpful discussions.

7. REFERENCES

[1] Chen, J., D. Ji, C.L. Tan, and Z. Niu. *Unsupervised Feature Selection for Relation Extraction*. in *IJCNLP-05*. 2005. Jeju Island, Korea.

[2] Dash, M. and H. Liu. *Feature Selection for Clustering*. in *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2000.

[3] Etzioni, O., M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates, *Unsupervised named-entity extraction from the Web: An experimental study.* Artificial Intelligence, 2005. **165**(1): p. 91-134.

[4] Feldman, R. and B. Rosenfeld. *Self-Supervised Relation Extraction from the Web*. in *ISMIS-2006*. Bari, Italy.

[5] Halkidi, M., Y. Batistakis, and M. Vazirgiannis, *On Clustering Validation Techniques.* Journal of Intelligent Information Systems, 2001(17:2/3): p. 107-145.

[6] Hasegawa, T., S. Sekine, and R. Grishman. *Discovering Relations among Named Entities from Large Corpora*. in *ACL 2004*.

[7] Lewis, D.D., Y. Yang, T. Rose, and F. Li, *RCV1: A New Benchmark Collection for Text Categorization Research.* Journal of Machine Learning Research, 2004. **5**: p. 361-397.

[8] Rosenfeld, B. and R. Feldman. *High-Performance Unsupervised Relation Extraction from Large Corpora*. in *ICDM-06, IEEE International Conference on Data Mining*. 2006. Hong Kong.

[9] Rosenfeld, B. and R. Feldman. *Using Corpus Statistics on Entities to Improve Semi-supervised Relation Extraction from the Web*. in *ACL-2007*.

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

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