CLASSIFYING ENTITY RELATIONS IN NATURAL LANGUAGE

An Honors Project

Presented by

Benjamin E. Lambert

Submitted June 2003

Guidance Committee Approval Signatures:

Andrew McCallum, Computer Science

David Jensen, Computer Science

CLASSIFYING ENTITY RELATIONS IN NATURAL LANGUAGE, Benjamin E. Lambert (Andrew McCallum, Guidance Committee Chair) Department of Computer Science, University of Massachusetts Amherst, Amherst, MA, 01003

In an effort to expand upon previous work, this research investigates the automation of classifying relations between entities (such as organizations, people and locations) in natural, written language. This research differs from earlier endeavors in the types of relations sought and the assumed information availability. More general relations, such as *located at* and *part of*, are sought on a per document basis with the assumption that complete knowledge of co-reference is available. To automate this task, three feature-based machine learning algorithms are used to train software classifiers. Using only relatively simple features, such as sentence tokens, digrams, and trigrams, many of these relations can be identified. The middling results achieved with few features show this task to have a potential and support further study.

Honors Project (499P)

Classifying Entity Relations in Natural Language

Benjamin E. Lambert

June 9, 2003

1 Introduction

Much of the information contained in natural, human language could be of great use if it were available to computer systems. Though recent advances in natural language processing have expanded the extent of the availability of this information, the incredible complexity of natural language leaves much of this information in a form not readily decipherable. Decoding natural language often requires the consideration a tremendous volume of data. Rather than using the data to manually compile knowledge bases and construct grammars, one might use machine learning to mitigate the tedium of doing so, if not surpass the practical limitations. Only in the last decade or so, with the increasing availability of annotated corpora of data and computer systems capable of handing the rich datasets, has machine learning become a very powerful tool for natural language processing (Lluís Màrquez, 2000).

Machine learning has been applied to many natural language tasks. Part of speech tagging, word sense disambiguation, context sensitive spell checking, grammatical parsing, and information extraction are among the many. All of these both seek and rely, to varying degrees, on information from two fundamental linguistic sources: syntax and semantics. Parsing, for instance, seeks a syntactical analysis, whereas information extraction is inherently semantic deduction. Most computations on natural language rely partially on both. As an example, word sense disambiguation is an instance of semantic inference, but will likely draw upon syntactical information, such as the word's part of speech, and semantic information, such as the word's definitions for that part of speech.

The research described in this paper examines the extraction of semantic information from natural language, specifically how the entities described in a written language source are related to each other. In other words, given two entities in a document, how are they related, if at all? Consider this sentence from the New York Times: "Israeli troops maintained their hold today on the Palestinian town of Beit Hanun, in the northern Gaza Strip." This sentence alone provides the reader with several pieces of information, including: Israeli troops maintained control of Beit Hanun, Beit Hanun is a Palestinian town, Beit Hanun is in the northern Gaza Strip. The information that can be described relationally is the focus of this research. The phrase "Beit Hanun is located within the Gaza strip" describes Beit Hanun in relation

to the Gaza Strip. In this case, the relation is of the type *located in*. The experiments described in this paper measure the extent to which machine learning may be applied to extracting this relational information.

In this study, relational information is extracted from newspaper articles, specifically New York Times articles and Associated Press reports. Though the relational information provided in any sample of language may be profuse, we focus on a few well-defined types of relations provided by the dataset. These relations may only exist between specific types of entities. The dataset defines relations between these fives types of entities: person, place, organization, facility, and global/political entity. The relations defined between these entities are: located near, located at, part of, role, and social. Examples of the latter two relations, role and social, would be an executive to a corporation and a pair of siblings, respectively. Results so far have shown that inferring these relations is difficult to achieve with high accuracy. The methods described in this paper were able to recall around half of the examples of each relation type and do so with approximately 50% precision. Though these numbers are not as good as one would generally desire to produce useful applications, they were achieved using only a small portion of the information available and consequently support the need for further study. Similar techniques have been able to infer more specific relations, such as that between a person and his or her birthplace, with greater accuracy (see related work, section 6).

Automating the classification of these relations has many direct and indirect applications. The relation between two entities in a sample of natural language is often one of the more pertinent facts contained within it (Three Mile Island is where?). The general task of identifying entities in natural language and classifying their relations also plays a key role in many endeavors that fall under the realm of information extraction and language understanding. Human-computer interaction may be one of the most useful applications of language understanding, whereby complex or ambiguous commands may be better interpreted. Direct applications of this research include data mining the described relational data. Specifically, this may include tracking an organization's changing locations or social network analysis to flag potential terrorist cells. Another application is identifying documents in which two entities are related to enhance searching for documents. For instance, one may search for documents that discuss the relation of two entities, rather than documents incidentally mentioning both.

This paper first formally describes the task in section 2. The experiments performed used both data that was provided (and so assumed to be attainable) as well as generated data; the methods for acquiring both types of data are detailed in section 3. Section 4 describes the machine learning algorithms used and the specific features from the data available to the algorithms. Section 5 presents the results, followed by sections describing related research and future research.

2 Formal Task Description

Relation classification may done at many levels. Zelenko et al. (2002) seeks to identify relations from individual sentences. That is, given a single sentence mentioning two entities, how are those entities related? Another approach, is to identify relations globally, utilizing information from an array of documents and databases. The research described in this paper attempts to classify relation examples in individual documents, specifically news articles.

Classifying relations in a multi-sentence document has one obvious advantage over a single-sentence classification: there is more information available. However, because much of this additional information may be extraneous, only the sentences that mention two entities are considered when classifying their relation. So, for any pair of entities in a document, all the sentences that mention both are considered. Thus where Zelenko et al. (2002) use a single sentence, we may use several. All co-reference resolution is assumed to have been done, so mentions may be proper names, pronouns, or other mentions. With the assumed co-reference, most related pairs of entities are mentioned together in several sentences for each document.

To more precisely describe the problem, entities, mentions and relations are formally defined as follows.

Definition 1 *An entity E_i is the i^{th} entity in a document, where E_i is designated by at least one noun or phrase.*

These entities are real-world manifestations; the document only mentions them. All entities in a given document are mentioned at least once and usually several times.

Definition 2 *A mention M_{ij} is the j^{th} word in the document that refers to the entity E_i.*

Though mentions provide the data necessary to retrieve clues about relations, the relations are defined between entities. Relations do not persist across documents. Relations are assumed to not persist because the data does not specifically list persistent relations, such as a city to a country. Other relations examined here are not necessarily persistent, such as the location of an organization, which may relocate, so no relations are considered persistent.

Definition 3 *A relation R_{xy} is the relation between two entities E_x and E_y.*

Here is an example of a pair of entities, examples mentions, and their defined relation: E_1 = *The White House*, E_2 = *Washington, D.C.*, M_{11} = "The White House", M_{21} = "Washington", R_{12} = *located at*. The dataset describes many such relations, but there are many other pairs of entities adhering to the constraints (i.e. mentioned in the same sentences) that do not have a defined relation. These pairs are assigned relations but their type is set to *unrelated*. These *unrelated* relations are the most common in the dataset, comprising approximately two-thirds of the qualified entity pairs. Thus there are six relation types: *located near*, *located at*, *part of*, *role*, *social*, and *unrelated*.

For each document, relations are created for every pair of entities mentioned in the same sentence at least once. Pairs of entities that are not mentioned in the same sentence at least once are ignored whether they have defined relations or not. Ignoring these pairs, relatively few of the defined relations (perhaps 5-10%) and a very large number of *unrelated* pairs (tens of thousands) are lost. The algorithm used to create lists of relations for each document is as follows:

> For all entity pairs E_i, E_j where $i \neq j$, if their respective mentions M_{ia} and M_{jb} are in the same sentence for some a and b, then create the relation R_{ij}. If the relation R_{ij} is defined in the data, set the type to the defined type, otherwise set the type to *unrelated*.

Many of these relations are not commutative, $R_{ij} \nrightarrow R_{ji}$. For example, the White House is located at Washington, D.C., but Washington, D.C. is <u>not</u> located at the White House. In this study, the order is not considered; if $R_{ij} =$ *located at*, then either i is located at j or j is located at i. Making the distinction between orderings may be useful or necessary in many cases as many of the relations studied are hierarchical. The distinction is ignored here for simplicity.

Being defined on a per-document basis, the software representation of each relation retains access to the original document and data the provided about the document, such as the location of each mention in the document and the entities' types. Only some of this available information is used. The information is used in the form of what we call feature functions or features for short. Feature functions are generally boolean

functions, often mapping a word or pattern to the boolean value of its presence in the document. The specific features used are described in section 4.1.

Disparate sets of relations are used for training and then testing software classifiers. In the training phase, a set of relations is given as input to a classifier trainer. The classifier trainer, using one of the machine learning algorithms described in section 4.2, trains a software classifier with each relation's features. This classifier, in turn, can take an unknown relation example and its features as input and output a prediction as to the type of relation.

Experiments are performed by first splitting the documents into two sets, then creating a set of relations from each. One set provides the input to the classifier trainer; the other set is used to test the resulting classifier. Although the actual relation types of the testing examples in are unknown to the classifier, they are known and used to check the classifier's predictions.

One hundred and thirty articles from The New York Times and the Associated Press news service were used as train and test documents. In the articles, there were approximately six-thousand pairs of entities mentioned in the same sentences, that is six-thousand distinct R_{xy}'s. Of these, approximately two-thousand pairs were engaged in one of the five defined relations and approximately four-thousand pairs did not have defined relations and were designated as *unrelated*.

3 Assumptions and Difficulties

In generating the sets of relations, several assumptions were made about the information available. For one, we assume that the entities in a document can be identified. In the real-world of non-annotated data, identifying the entities cannot be taken for granted. This and other assumptions, such as identifying an entity's type, as well as how these tasks could be automated, are described in this section. Also described are the solutions used to gather additional data that was not provided, such as sentence boundaries.

3.1 Mention Identification

To find relations in a raw document, one first needs to identify the entity mentions. Proper name mentions are often found easily, for example, when preceded by titles, such as Mrs. or Dr. In a properly styled English text, sequences of capitalized words are usually proper names. In the less structured environment of the World Wide Web words are frequently capitalized for other reasons, such as a hyperlink to "World News." On news websites, many strings of capitalized letters can be identified to not be entity names if they are hyperlinks.

One complication to identifying proper names is locating the extent of a name. An example of this difficulty occurs when an organization's location is part of its name. Should the words "City University of New York" be treated as one institution or one

institution and a city? Research has shown that finding the extent of a proper names can be automated with fairly high accuracy (Wacholder et al., 1997). This problem is assumed to be solved as data is annotated with all mentions and their extents.

Although proper names are relatively easily identified, other types of entity mentions are not. A facility, for example, may be referred to only as "the administration building." When also referred to by a proper names, it is likely to be very difficult to resolve co-reference for these types of mentions. These mentions are assumed known, as well as their co-reference. This may be a rather large assumption, and its automation requires further study.

One type of mention that is easily found, is the pronoun. These can be looked up in tables. Unfortunately, a pronoun is practically useless without knowing its antecedent. Pronoun co-reference resolution is also difficult to automate but is assumed to have been done.

3.2 Entity Identification or Co-Reference Resolution

Working at the document level, a list of all mentions is not very useful by itself as many may refer to the same entities. What is really needed is a list of the entities and their corresponding mentions. Given all the mentions, finding distinct entities in a document is equivalent to resolving the co-reference. Although this resolution is assumed to have been done by this study, in many cases it is difficult to automate.

Resolving co-references among proper names is generally easier than among pronouns, but is very difficult in some cases. For example, the proper names "George W. Bush," "George H. W. Bush," and "Mrs. Bush" all refer to distinct people. Likewise, "President Bush" and "former President Bush" refer to distinct people. The gender implied by first names or titles often helps to distinguish the names. In cases such as the latter example given here, the distinction may be difficult if not impossible to make without additional information. Recognizing which names refer to which entities is especially important when looking for their relations; the importance can be seen in the preceding examples where all parties are engaged in *social* relations. Without the machinery to identify these as distinct entities, inferring meaningful relations among them is impossible.

The annotated dataset provided the enticing opportunity to not only use proper name mentions, but all referents. Not only can we assume that there is no need to disambiguate to whom "Mr. Clinton" and "Bill Clinton" refer, but there is also no need to resolve pronoun and other co-references. Unlike much of the other information assumed by this study, this information cannot be discovered automatically with very high accuracy. This research assumes the availability of all co-references. Morton (2000) provides a good description of co-reference resolution for similar tasks. With the ability to accurately locate all entity mentions and resolve their co-reference, lists

of all entities along with all of their corresponding referents can be created. These lists are provided by the dataset, making their deduction unnecessary and assumed.

Although automating pronoun co-reference with useful accuracy is likely to add significant computational overhead, it is very compelling. Even if co-references cannot be resolved with high accuracy, the lack of any co-reference data will render some relations impossible to identify. Sentences explicitly describing a relation between two entities often refer to at least one of the two entities with a pronoun. This is not surprising as sentences with many proper names are often cumbersome. Although knowledge of pronouns and other referents potentially allows more relations to be discovered, if co-reference resolutions are not highly accurate they may degrade performance. Imperfect co-reference resolution might be used more effectively by including a confidence measure, where data provided by tenuous pronoun resolutions is weighted less heavily.

3.3 Entity Type Identification

One piece of information that can often be acquired, and is very powerful, is an entity's type. Related research (Roth and tau Yih, 2002), detailed in the related work section of this paper, demonstrates how deeply intertwined relation and entity types are. In essence, this is due to the fact that most relations can only exist between certain types of entities. For instance, a social relation can only be defined between people.

The dataset used provides all entity types, so this is assumed to be known. Generally, acquiring this information is a bit more difficult than a table lookup, though it can be done to an extent.

Some entity types are easily identified. For instance, entity names beginning with a title, such as Mr. or Mrs., almost always refer to people (there are always exceptions, such as "Mr. Tux"). Proper names ending with strings like "Corporation" or "Ltd." likely designate organizations. Other names require the use of context or global knowledge to identify type and are consequently difficult to identify computationally. An example of this is the name "Hank Hill," which may be a person or place. Wacholder et al. (1997) use the "Nominator" system to correctly identify 99% of entity types after discarding 21% of low confidence decisions.

3.4 Language Structure

Many natural language processing approaches treat data as an unstructured collection of words. These generally focus on word presence, frequency, and combination; this is often called the "bag of words" approach. The research described in this paper utilizes the structure of each document, in particular the word order and sentence boundaries. Although not included in this study, grammatical sentence parses may be one of the richer sources of structural information. Like co-reference resolution, parsing is difficult and usually cannot be done with very high accuracy.

Sentence boundaries are used in a few ways; one is to determine whether two entities are mentioned in the same sentence. They are also used when creating features from the text, as features are only created from sentences that mention both entities (as opposed to from the entire document). In an appropriately styled English document with two spaces between sentences, locating sentence boundaries is trivial. Data used in this research, unfortunately, separated sentences with only one space. Fortunately, of all the necessary prerequisites, finding these sentences boundaries may be the easiest.

Every sentence boundary in the dataset is of the form: period, white space, capital letter. The non-triviality of finding sentence boundaries is due to the presence of this pattern mid-sentence. This usually occurs when a proper name follows an abbreviation. A few simple rules identified nearly all these instances. One rule that catches almost all false sentence boundaries is that sentences cannot end with a title or single capitalized letter (as in a middle name).

4 Methodology

To classify relations, we use relation examples from half of the documents to train software classifiers. These classifiers are essentially functions that map sets of features to relation types. Features may describe the relation example in many ways; for example, a feature may attest to the presence of a word in a sentence. Once trained,

a classifier takes a set of features, a feature vector, as input and outputs a predicted relation type. The features used to describe each relation example are described in section 4.1. A brief description of each of the learning algorithms used to train the classifiers is given in section 4.2.

4.1 Features Used

4.1.1 Entity Type

Perhaps one of the most basic, yet powerful feature types is the entity type. For each of the entities in a relation pair, a feature is added to the feature vector denoting the entity's type. For instance, given a pair of entities where one is a location and the other is a facility, the features "LOCATION" and "FACILITY" are added to the feature vector. Another feature is added to describe composition of the pair. In this case, the lexicographically later entity type is appended to the other (as we are ignoring the order); this feature describes the conjunction of the two types.

4.1.2 The "Bag of Words"

Another basic feature type used is the set of tokens present in sentences mentioning both entities. Given an entity pair, all sentences that mention both entities (there may be several) are retrieved from the sentence file. Each of these sentences is tokenized, or broken into individual words, and each token becomes a feature. Formally, given E_i and E_j, tokenize all sentences that contain both M_{ia} and M_{jb}, for some a and b,

and add each token to the feature vector. These features do not capture any of the sentence structure, but in many cases provide valuable information. For instance, the mere presence of the word "brother" may indicate a social relation, specifically a fraternal one. This feature is used slightly differently from the traditional "bag of words" approach, where all words in the entire document are used.

Another difference from many "bag of words" approaches, is that stopwords are not removed. The stopwords removed are generally a few hundred of the most common English words, such as articles and prepositions. When looking for relations however, ignoring these words may be detrimental. For example, one of the relations sought is the *located at* relation. For this relation, the stopword "at" may be crucial to identifying it. One experiment of interest would be to determine how important these stopwords are to classifying relations like *at* as their high frequency will limit their usefulness.

Another common technique used in natural language processing, not used here, is stemming. Stemming involves removing common word endings, so that the tense does not interfere with the semantics. For example, rather than use the word "learning" as a feature, stemming would remove the ending and use the feature "learn".

4.1.3 Digrams, Trigrams and N-Grams

Digrams, trigrams and n-grams are similar to sentence tokens, though they begin to capture the word order. Digrams are pairs of adjacent words. Formally, for every

word in a sentence (besides the last), a space and the succeeding word are appended to create the digrams. Similarly, trigrams are 3-tuples of adjacent words, created from all sequences of three consecutive words. Digrams and trigrams are only created from the sentences mentioning both entities in each relation example. Features could also be created from all n consecutive words, with n greater than three, called n-grams; experiments showed that 4-grams and 5-grams were not useful, so they were not used.

Creating digrams and trigrams directly from the sentences does not capture many otherwise useful patterns. These patterns are formed when similar entities occupy the same role in a sentence. To put the patterns to work, all mentions of the two entities in their mentioning sentences are replaced by their corresponding entity types. For example, the clause, "UMass is located in Amherst," would become, "ORGANIZATION is located in LOCATION." The trigram "located in LOCATION" will more often assist with relation classification than the specific, "located in Amherst" which will only help find entities that are located in Amherst.

4.2 Machine Learning Algorithms

Using all the features listed above, one might attempt to manually construct a set of rules for determining the relation of an entity pair. However, considering that the presence of individual words or combinations of words may constitute a rule and that there are around 100,000 or so English words, this may not be practical. Feeding the

features to a machine learning algorithm may save a good deal of time and may also pick up on patterns that would have otherwise gone unnoticed.

Three algorithms were used to train classifiers with the features described in the previous section. The algorithms were: naive Bayes, maximum entropy, and winnow. Although each algorithm displayed strengths and weaknesses, maximum entropy generally appeared most successful. The MALLET (MAchine Learning for LanguagE Toolkit) (McCallum, 2002) implementations were used for all algorithms. Each algorithm is described briefly in the following sections.

4.2.1 Naive Bayes

Given an unknown relation R, with a set of features, x = $< x_1, x_2, ..., x_n >$, we want to compute the probability that relation R is of each known type of relation based on the training examples. If we assume that the probabilities of each feature occurring are independent, Bayes' theorem allows us to compute this probability. That is, we can compute the posterior probability that R is some relation, say r (where r=*located at*, *part of*, etc.), $P(r|x)$, from the conditional probabilities that given relation r, the example will have the feature x_i, $P(x_i|r)$. Bayes' theorem defines this probability as:

$$P(R|x) = \frac{P(x|R)P(R)}{P(x)}$$

The naive Bayes algorithm is naive because of its incorrect assumption that the probabilities of each feature are independent. They are not independent in this case; for example, articles depend on their nouns. The probabilities must be independent to calculate the probability of set of features all occurring. The probability of this conjunction of events is only equal to the product of their probabilities if the events are independent. Despite the incorrect assumption, this algorithm often performs comparable to other learning algorithms. Naive Bayes computes the probability that R is of relation type, r, given the set of features, x, as:

$$P(R = r|x) = \prod_{i=1}^{n} P(x_i|R = r)$$

The training phase of this algorithm calculates the conditional probabilities $P(x_i|r)$ for each relation type. The classifier then easily computes this product to make the classification. The classifier will ignore features that it has never seen before (essentially words not in its lexicon), as it knows nothing of their probabilities. The MALLET implementation of naive Bayes uses a slightly more sophisticated algorithm to ensure that in the case that $P(x_i|r) = 0$ for some i, the entire product is not zero. This is described in detail in McCallum and Nigam (1998).

4.2.2 Maximum Entropy

As implied by the name, this algorithm seeks to maximize the entropy of its classifications. As a simple example, given no information with which to classify relations, a maximum entropy classifier will classify each relation type of the six with 16.7% chance. Fortunately, the training data provides information to make more rational decisions. Given the statistic that 50% of all training pairs mentioned in sentences containing the word "at" are of the type *at*, the classifier obeys this statistic when making classifications and makes all other decisions with maximum entropy. With this statistic, or constraint, 50% of relation pairs mentioned in sentences containing the word "at" will be classified as *at*, the remaining 50% will be classified as one of the other relations with 20% chance. The relation pairs mentioned in sentences not containing the word "at" will be classified to each relation with equal probability, 16.7%.

To enforce these constraints, the maximum entropy algorithm ensures that the expected value of each feature in the training data is consistent with the classifications made on the testing data. So the distribution of $P(r|x)$, where x is a relation example, must be consistent with this constraint:

$$\frac{1}{|T|}\sum_{x\in T} f_i(x, r(x)) = \frac{1}{|T|}\sum_{x\in T}\sum_{r} P(r|x) f_i(x, r)$$

where T is the set of training examples, $r(x)$ is the relation type of example x, and $f_i(x, r)$ is a the value of relation r's i^{th} feature for the relation example x.

Modeling the constraints in this way we are guaranteed a unique distribution to classify testing examples. It is also the case that this distribution is exponential (Pietra et al., 1997). The distribution is described by the following exponential distribution function:

$$P(r|x) = \frac{1}{Z(x)} exp(\sum_i \lambda_i f_i(x, r))$$

where λ_i are the parameters associated with each feature and $Z(x)$ is a normalizing factor defined as:

$$Z(x) = \sum_r exp(\sum_i \lambda_i f_i(x, r))$$

The training stage of the maximum entropy algorithm calculates the optimal set of parameters λ_i. Although the details are beyond the scope of this paper, the search space has a unique maximum likelihood and no local maxima. Therefore, the optimal set of parameters can be found by hill-climbing beginning from an arbitrary point. The MALLET implementation of maximum entropy uses improved iterative scaling to hill-climb and Gaussian priors to reduce over-fitting. Details on these two techniques are described in Nigam et al. (1999).

4.2.3 Winnow

The winnow algorithm uses mistake-driven multiplicative update to calculate weights for each feature in a linear separator function. The algorithm takes three parameters: a threshold θ, a constant promotion multiplier α, and a constant demotion divisor β. The algorithm is mistake-driven in that it only updates the weights when a relation is improperly classified. During training, the algorithm gives a yes/no decision on each training example for each relation type (so it may say yes to multiple relations types). Each relation type has a set of weights, w_1, w_2, ..., w_n, corresponding to each feature, which are initially all set to one. A training example is claimed to be the relation corresponding to the weights if the following inequality is true:

$$\sum_{i=0}^{n}(w_i x_i) > \theta$$

If this inequality correctly identifies the relation, nothing happens. If it incorrectly makes the affirmative decision, then the weight of each feature corresponding to the training example is demoted via division by the constant parameter β. If the algorithm fails to identify the correct relation, the weights of each feature pertaining to that example are promoted via multiplication by the constant parameter α. For the experiments described in this paper, parameters were set to $\alpha = 2$, $\beta = 2$, and $\theta = n/2$, where n is the number of distinct features for a relation's training examples.

Winnow may be used as an online algorithm, continuously updating the weights as each test example is classified. For testing purposes, all weights remain constant after the training phase. After training, each test example is classified by calculating the above sum for each relation type; the relation type with the greatest sum, whether or not is is greater than the threshold, is returned as the predicted relation.

5 Experiments

To measure success with classification, precision and recall are used, as accuracy alone does not describe the success well. Assuming that finding related pairs is more important than finding unrelated pairs, we should be more concerned with how many and how precisely we can find related pairs, than how accurately we can find unrelated pairs. Recall and precision measure those two quantities, how many and how precisely, and f-measure is an inverted average of the two.

$$Precision(T) = \frac{\text{Number of pairs correctly classified as type T}}{\text{Number of pairs classified as type T}}$$

$$Recall(T) = \frac{\text{Number of pairs correctly classified as type T}}{\text{Number of pairs of type T}}$$

$$F - Measure(T) = (\frac{\frac{1}{a} + \frac{1}{b}}{2})^{-1}$$

$$Accuracy = \frac{\text{Total number of pairs correctly classified}}{\text{Total number of pairs}}$$

Because the dataset contains so many entity pairs that are *unrelated*, any classifier may be compelled to classify all pairs as *unrelated*. In doing so, it would on average be about 68% accurate. Randomly classifying relations with probability equal to their empirical distribution in the training set would achieve approximately 48% accuracy on average. Classifying in this way, proportional to the empirical distribution, one would expect the recall and precision for each relation type to be equal to its frequency in the training set. For example, *role* relations constitute approximately 14% of the examples. If making the *role* classification arbitrarily 14% of the time, we should statistically be right 14% of the time. Likewise, the recall would on average be proportional to the frequency in the training data.

	Part	Role	At	Near	Social	Unrelated
Percentage (%)	4	14	10	1	3	68

Table 1: Approximate distribution of relation types

If we are indeed more interested in the related examples than the unrelated ones, the frequency of each relation (shown in table 1) serves as a good absolute minimum to beat. In other words, if we cannot achieve 14% precision and recall for *role* relations then we may as well just classify randomly.

Another number we would like to beat for each relation type, is the proportion of that relation to all related types. Of all the defined relations, about a third of them are *at* relations. The extent to which this number can be surpassed in the precision and recall of each relation is an indicator of how well related pairs can be distinguished among the five relations, as opposed to being identified as only *related* and then randomly assigned a specific relation.

It is also worth noting how few of the relations are of the type *near*. With so little training data, classifiers perform poorly on this relation and rarely classify any relations as *near*. Results for classification of *unrelated* pairs are also included, but may also not be of great interest.

5.1 Success of individual features

To judge which individual types of features may be most relevant to the classification, experiments were initially isolated to individual feature types. Some features prove to be quite powerful when used alone.

5.1.1 Entity Type

The correlation between entity type and relation type, as can be seen in (Roth and tau Yih, 2002), was evident in the results of only using the entity type feature (table 2). Of the three algorithms, using only this feature, naive Bayes is best able to find related pairs. Although naive Bayes identifies related pairs best, its accuracy is

lowest because it predicts many *unrelated* relations to be related, whereas the other two almost always predict *unrelated*. Of features most correlated with relation types, a few of the best are: "PERSON" and "PERSON-PERSON", which are both correlated with *social* and *role* relations; "GPE-GPE" which is highly correlated with *at*; and "PERSON-ORGANIZATION" which is highly correlated with *role*. The "PERSON-PERSON" feature allows naive Bayes to recall 100% of the *social* relations.

	Naive Bayes			Maximum Entropy			Winnow		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Part	12.9	6.11	8.29	44.44	3.05	5.71	44.44	3.05	5.71
Role	38.28	35.96	37.08	0	0	0	0	0	0
At	56.86	31.41	40.46	52.78	6.86	12.14	0	0	0
Near	25	14.29	18.18	0	0	0	0	0	0
Social	18.09	100	30.64	0	0	0	0	0	0
Unrelated	75.55	75.18	75.36	71.28	99.2	82.96	70.88	99.78	82.88
Accuracy	63.04			71			70.81		

Table 2: Precision, recall, and f-measure using only entity type features

5.1.2 Sentence Tokens

Tokens alone in some, specific cases can provide strong evidence of a relation. For example, one of the most correlated of these features was "kilometer", which was highly correlated with both *at* and *near*. After "kilometer" in relevance was the feature "at". Though this feature is probably most correlated with the *at* relation, it is generally a very common feature. This correlation would seem to justify the decision to include stop words as features. In this experiment, results shown in table

3, naive Bayes sacrificed recalling *unrelated* pairs almost entirely (20.91%), in favor of related pairs. This may be because related pairs are more likely to have words in common than unrelated pairs. That is, there are likely to be common features, such as "at", which boost the probabilities of the defined relations. Whereas *unrelated* relations are less likely to have similar words and so their features will tend to have lower probabilities (many $P(x_i|r = unrelated)$ with low values). The product of these lower probabilities will be especially low, and so deemed an unlikely classification.

	Naive Bayes			Maximum Entropy			Winnow		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Part	3.9	15.27	6.21	16.67	0.76	1.46	33.33	0.76	1.49
Role	19.44	48.54	27.76	20.24	23.15	21.59	26.19	2.47	4.52
At	15.08	46.21	22.74	19.18	10.11	13.24	20	1.08	2.05
Near	0	0	0	0	0	0	0	0	0
Social	15.52	16.98	16.22	28.57	11.32	16.22	0	0	0
Unrelated	73.02	20.91	32.52	72.65	80.55	76.40	70.92	94.63	81.07
Accuracy	26.55			61.4			67.51		

Table 3: Precision, recall, and f-measure using only token features

The two most correlated token features, that is the two most closely corresponding to a relation type, were the strings "(AP)" and "␣". These two strings lead off every article from the Associated Press preceded by the location of the story. These tokens are treated as part of the first sentence. Examining the results showed that one reason this is useful is that entities mentioned in the first sentence are often unrelated. Another reason is that entities mentioned in the first sentence are often located at the

location of the story (which leads off the first sentence). These features are an example of some of the very useful features resulting from a similar style in the documents. Features based on a common document style will not help when classifying relations in another domain, and so have limited use.

Other features that were highly correlated to relations were frequently mentioned places, such as France and the U.S. The words "where", "friend", and "near" also appear to be fairly well correlated to *at*, *social*, and *near* respectively.

5.1.3 Digrams

Digrams tend to be more useful than the raw set of tokens, as they are able to carry information about the word order. Using only digrams, the classifier is able to identify a fair number of relations. Results are shown in table 4. A few of the most relevant digrams are: "GPE, GPE", which is very highly correlated to *at*, "Boston, MA" for example; "GPE PERSON", which is correlated with *role*, as in "British diplomat"; and "ORGANIZATION PERSON" which is correlated with *role*, such as "Commerce Ministry official".

With digrams, naive Bayes continues to sacrifice the identification of *unrelated* pairs in favor of recalling related pairs. Naive Bayes does not indiscriminately choose related types, but has some heuristics toward which relation. This can be seen in the proportionately high recalls for *part*, *role*, *at*, and *social*. Although entity type features are not used directly in this experiment, entity mentions are replaced with their types

in the digrams; this may help explain the very high recall for *social* relations with naive Bayes. As opposed to naive Bayes, maximum entropy is able to retain over 70% precision without exclusively predicting *unrelated*.

	Naive Bayes			Maximum Entropy			Winnow		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Part	8.41	62.6	14.83	43.69	34.35	38.46	21.31	19.85	20.55
Role	34.92	36.18	35.54	51.92	33.48	40.71	34.01	22.7	27.22
At	18.73	46.93	26.78	34.59	23.1	27.71	19.64	11.91	14.83
Near	0.51	9.52	0.97	0	0	0	0	0	0
Social	7.05	73.58	12.87	35	13.21	19.18	3	5.66	3.92
Unrelated	78.3	3.69	7.04	77.94	89.43	83.29	73.8	81.66	77.53
Accuracy	15.63			71.69			62.98		

Table 4: Precision, recall, and f-measure using only digram features

5.1.4 Trigrams

Although trigrams are able to capture more complex patterns than digrams, they do not generalize as easily. For example, the trigrams "PERSON of the", "in the GPE" are highly indicative of *role* and *at*, respectively. However, there are relatively few such features in the fairly small data set. The utility of digrams can be seen by the pertinence of the trigram "GPE PERSON and", which only occurs a few times, but fills in for the absent digram "GPE PERSON". Only maximum entropy faired quite well with trigrams only, sporting a high accuracy and a few very high precisions. The high precisions seem to be due to the accuracy of very specific features, such as "PERSON of the".

	Naive Bayes			Maximum Entropy			Winnow		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Part	11.07	46.56	17.89	79.17	29.01	42.46	4.08	26.72	7.08
Role	48.94	5.17	9.35	61.63	11.91	19.96	31.23	18.88	23.53
At	26.95	13.72	18.18	90	6.5	12.12	28.71	10.47	15.34
Near	0.71	71.43	1.41	0	0	0	0	0	0
Social	7.94	37.74	13.11	0	0	0	22.73	9.43	13.33
Unrelated	79.01	2.84	5.49	73.45	98.67	84.21	72.39	61.94	66.76
Accuracy	6.95			73.32			48.69		

Table 5: Precision, recall, and f-measure using only trigram features

5.2 Combinations of Features

Across the board, nearly every combination of feature types benefits when the entity type feature is included. Given the apparent importance of entity type, it is not surprising that it helps the others. Table 6 compares the results of using digrams alone (one of the best single features) with the combination of digrams and entity type. Although winnow fell back to always predicting *unrelated*, its overall accuracy increased. Both naive Bayes and maximum entropy benefit on almost every measure when entity types are included.

Overall, the results when using multiple feature types depend more on the learning algorithm used than the specific set of features. For example using maximum entropy with trigrams and entity types yields very similar results to using maximum entropy with tokens, digrams, trigrams and type. Winnow and naive Bayes also performed relatively consistently with different sets of features. However, the three

	Naive Bayes		Maximum Entropy		Winnow	
	Dig	Dig+Typ	Dig	Dig+Typ	Dig	Dig+Typ
Part Prec	8.41	8.49	43.69	44.07	21.31	0
Part Rec	62.6	67.94	34.35	39.69	19.85	0
Role Prec	34.92	33.47	51.92	50.53	34.01	0
Role Rec	36.18	56.4	33.48	42.92	22.7	0
At Prec	18.73	22.01	34.59	51.71	19.64	0
At Rec	46.93	71.12	23.1	43.68	1.91	0
Near Prec	0.51	2	0	0	0	0
Near Rec	9.52	4.76	0	0	0	0
Social Prec	7.05	15.86	35	42.22	3	0
Social Rec	73.58	86.79	13.21	35.85	5.66	0
Unrelated Prec	78.3	80.14	77.94	80.4	73.8	70.84
Unrelated Rec	3.69	5.2	89.43	85.79	81.66	100
Accuracy	15.63	22.05	71.69	72.82	62.98	70.84

Table 6: Precision, recall, and f-measure with digrams compared to digrams and entity types

algorithms performed very different from each other. The next section compares the three algorithms.

Although, each combination of features had strengths and weaknesses, the best by accuracy was tokens, digrams, trigrams, and types (all the available features) in conjunction with the maximum entropy algorithm (table 7). This combination classified relations with 74.87% accuracy.

To better illustrate what these numbers mean, the confusion matrix in table 8 shows how each relation example was classified. The prediction made for each test example is shown on the left key, and the actual relation type is shown on the top key. Most predictions for one of the five defined relations were either correct or were

	Part	Role	At	Near	Social	Unrelated
Precision	55.17	56.21	56.11	0	55.17	79.72
Recall	36.64	44.72	36.46	0	30.19	89.52
F1	44.04	49.81	44.2	0	39.02	84.33

Table 7: Precision, recall, and f-measure with tokens, digrams, trigrams, and type with the maximum entropy algorithm.

really *unrelated*. This shows how well maximum entropy is able to distinguish among the relations. The fact that most of the mistakes made by maximum entropy were *unrelated* examples classified as related and vice versa, shows that its main difficulty is with determining if any relation exists.

	Part	Role	At	Near	Social	Unrelated
Part	48	9	3	1	0	26
Role	1	199	14	0	0	140
At	7	5	101	5	0	62
Near	0	0	0	0	0	0
Social	0	5	0	0	16	8
Unrelated	75	227	159	15	37	2016

Table 8: Confusion matrix for tokens, digrams, trigrams, and type with the maximum entropy algorithm.

5.3 Analysis of Algorithms

As mentioned in the previous section, multi-feature experiments tend to depend far more on the algorithm than the feature set. Table 9 shows the average numbers

achieved from multi-feature experiments for each learning algorithm. Results from each of the experiments were generally within a few percentage points of the averages.

	Naive Bayes	Maximum Entropy	Winnow
Part Precision	8.81	53	45
Part Recall	49.43	36.64	1.72
Role Precision	28.65	52.32	9.18
Role Recall	64.1	43.6	1.01
At Precision	21.23	55.1	3.57
At Recall	68.77	37.09	0.63
Near Precision	0.08	50	0
Near Recall	1.19	2.38	0
Social Precision	20.46	44.51	0
Social Recall	54.25	28.3	0
Unrelated Precision	79.01	79.54	70.9
Unrelated Recall	5.42	88	99.19
Accuracy	21.75	73.67	70.53

Table 9: Average results for all multi-feature experiments for each algorithm

Naive Bayes nearly always thwarts the *unrelated* relation. Winnow nearly always predicts the *unrelated* relation. Maximum entropy consistently achieves around 50% precision for all defined relations (excluding *near*) and recall around 40%, while retaining an accuracy greater than 70%.

The confusion matrix in table 8 shows how the maximum entropy algorithm often predicted the right relation if the pair was indeed related in some way, but had difficulty differentiating between related and unrelated. This indicates that identifying which relation is generally not as hard as identifying the existence of a relation. The naive Bayes results support this. When rarely predicting a relation to be *unrelated*

it is expected that precisions will be so low, but the recall for the defined relations is disproportionately high with naive Bayes.

The inability of the winnow algorithm to identify many of the defined relations and defaulting to *unrelated* is likely due to the fact that it is mistake driven. Because there are so many more unrelated pairs than related pairs, affirmative predictions for the defined relations are likely to be wrong. As a consequence, weights associated with the defined relations will be rapidly demoted and the algorithm will refrain from choosing these relations. The winnow implementation used here is better suited to deal with only two classes (divided by the threshold); a system such as Dan Roth's Sparse Network of Winnows (SNoW) would be better suited to deal with the six. Another potential problem with winnow, is that the function may not be linearly separable.

6 Related Work

Much of the other work that has been done on classifying relations in natural language has focused on identifying very specific relations, opposed to the more general relations we sought (i.e. *role* and *social*). The more focused approach appears better able to hone in on specific linguistic patterns characteristic of a relation. Many of the correct classifications in this study may be due to the algorithm finding specific relations that are unfortunately placed in an umbrella relation type. For instance, one relation

studied by Zelenko et al. (2002), *person affiliation*, the affiliation of a person to an organization is, for better or worse, placed all these relations into the much more general *role* relation. One advantage to focusing on specific relations is that the entity pairs can be filtered beforehand. Because the *person affiliation* relation is only defined between a person and an organization, only pairs of these entity types need to be considered.

Zelenko et al. (2002) also studies the *organization location* relation. For both of these specific relations studied, precision approached 90% and recall approached 80% with approximately 1,000-2,000 training examples. They use kernel functions on shallow parse trees containing node attributes. Experiments performed in Zelenko et al. (2002) differed from those described in this paper in that entity pairs were first filtered, only one relation was examined at time (as opposed to five), and sentence-based parse features were used with kernel methods.

The use of kernel methods largely captures the relative similarities of two parse trees through common subtrees in the trees. Zelenko et al. (2002) utilize the fact that shallow parse trees identify the key parts of a sentence, rather than all the intricacies of its syntax as a full parse does. Parse trees were annotated with each entity's role in the relation and the kernel functions judged the similarities among examples. Their experiments, using these techniques with SVMs and Voted Perceptrons,

showed slightly better performance and a steeper learning curve than feature-based experiments with Naive Bayes and Winnow.

Preliminary research, with a slightly different approach, by Roth and tau Yih (2002) showed promising results for relation recognition. This research combines the problems of identifying entities types and identifying relations into one larger problem. One motivation for this approach was due to erroneous named-entity tagger decisions propagating into the relation identification. For instance, tagging "Chelsea", the Massachusetts town, as a person may have a large impact on results because entity type is of central importance to relation classification.

The two very specific relations examined in Roth and tau Yih (2002) are *killed* and *born in*. The *killed* relation occurring between two people when one person kills the other, and *born in* being the relation of a person to his or her birthplace. To classify the entity types and relation, the probabilities of each entity and relation example being of a specific type (person, place, kill, or born) are first computed independently based on several features (nearly the same features mentioned in section 4.1). With these probabilities, the relations and entities are assembled into a two-layer belief network, one layer for the entities, another for the relations. Each relation connects to two entities, as they are binary relations. Finding the most probable class assignments is equivalent to finding assignments for all the variables in the network that maximizes the joint probability. Although this most-likely-explanation problem is intractable in

this situation (because its not acyclic), Pearl's belief propagation algorithm applied iteratively empirically generates good results (Murphy et al., 1999). It is worth noting that the precision and recall achieved in this research, though very promising, were not as good as in Kernel Methods (Zelenko et al., 2002), especially the recall for *kill* relations which was generally 50% or less.

Although very different in goal, Paul Ogilvie's May 2000 University of Massachusetts honors thesis (Paul Ogilvie, 2000) presents the use of linked-object representations for topic tracking. The linked-object representation is highly analogous to entity-pairings used in this research. These simple data structures appear to be quite powerful for representing the more salient information in natural language text. The topic of an article is in many ways one of the most important and relevant aspects of it. Among Ogilvie's findings, was that naive co-reference resolution – associating pronouns with the proper name in the preceding sentence, if one is present – did not improve topic-tracking abilities.

7 Future work

As noted in the introduction, natural language processing often uses both syntactic and semantic information. Much of the information gathered about relations in this research beyond the "bag of words" was purely syntactic. One frequently used technique that captures some of the semantic implications is the use of hyponyms,

hypernyms, and synonyms. These may extend the ability of certain features to generalize; for example, the feature "parent" may prove more useful if used with the additional hyponyms "father" and "mother." Such related words may be retrieved from WordNet (Fellbaum, 1998); WordNet features are used in Roth and tau Yih (2002).

Although digrams and trigrams capture information about the word order, and consequently some syntactic information, English syntax is not linear. Related work has used dependency parsers and shallow parsers (Zelenko et al., 2002) to examine syntax. Zelenko et al. (2002) found relative depth in the shallow parse to be particularly useful. Cultivation of full parse trees could provide tremendous additional information. For example, if one entity is mentioned in a prepositional phrase, the preposition might be very useful (consider the preposition "at").

Another extension to this research, and in fact how this research began, is to utilize web structure and web crawling. Craven et al. (1998) use web page content as well as hyperlink structure to extract social relations, such as that between a student and a professor. Their research focuses on the domain of university web sites. The same topological information may aid relation extraction in a more general web setting. For instance, on news web sites, pages linked in referential ways (as opposed to general navigation links) may provide additional information about relations.

Research described in this paper only scratches the surface of what one might try when attempting to classify relations. As mentioned before, this work investigates only a few concrete, yet very general, types of relations. One question to ask, is how does the generality of the relations sought affect the results? Are very specific relations always easier to identify? And which one are the easiest?

8 Conclusion

This research attempts to expand on earlier research into the problem of classifying relations. It explores this task with a more general perspective, with the goal of learning less specific and more conceptual information from written documents. Though not as many as desired, many relations are correctly classified with these methods. These successes, achieved using relatively little information, show that there is potential and support the need for further study. This research also exposes the proclivities of the three learning algorithms when used on this data and shows maximum entropy to be most effective.

The ability to classify relations in natural language has many direct applications and is a key to many, broader, related tasks such as language understanding. Success in related and future research could have widespread implications, from improving search engines to making computer interaction more tolerable to users. As computers

become more and more prevalent, bridging the gap between human language and computer language will become increasingly important.

References

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew K. McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US, 1998. AAAI Press, Menlo Park, US.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998. ISBN 0-262-06197-X.

Lluís Màrquez. Machine Learning and Natural Language Processing. Technical Report LSI-00-45-R, Departament de Llenguatges i Sistemes Informàtics (LSI), Universitat Politecnica de Catalunya, Barcelona, Spain, 2000.

A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

Andrew Kachites McCallum. MALLET: A Machine Learning for Language Toolkit. http://www.cs.umass.edu/ mccallum/mallet, 2002.

T. Morton. Coreference for NLP Applications. In *Proceedings ACL-2000*, 2000.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

K. Nigam, J. Lafferty, and A. McCallum. Using Maximum Entropy for Text Classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

Paul Ogilvie. Extracting and Using Relationships Found in Text for Topic Tracking. University of Massachusetts Honor's Thesis, May 2000.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing Features of Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), 1997.

Dan Roth and Wen tau Yih. Probabilistic Reasoning for Entity and Relation Recognition. In *COLING*, August 2002.

N. Wacholder, Y. Ravin, and M. Choi. Disambiguation of Proper Names in Text. In *Proceedings of Fifth Conference on Applied Natural Language Processing*, pages 202–208, 1997.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel Methods for Relation Extraction. In *Proceedings of the Conference on Empirical Methods in Natural*

Language Processing (EMNLP), pages 71–78, Philadelphia, July 2002. Association for Computational Linguistics.