Disambiguating Conjunctions in Named Entities

Paweł MAZUR
Institute of Applied Informatics
Wrocław University of Technology
Wyb. Wyspiańskiego 27
50-370 Wrocław,
Poland
Pawel.Mazur@pwr.wroc.pl

Robert DALE
Centre for Language Technology
Macquarie University
NSW 2109,
Sydney,
Australia
Robert.Dale@mq.edu.au

Abstract

The recognition of named entities is now a well-developed area, with a range of symbolic and machine learning techniques that deliver high accuracy identification and categorisation of a variety of entity types. However, there are still some named entity phenomena that present problems for existing techniques; in particular, relatively little work has explored the disambiguation of conjunctions appearing in candidate named entity strings. We demonstrate that there are in fact four distinct uses of conjunctions in the context of named entities; we present the results of some experiments using machine-learned classifiers to disambiguate the different uses of the conjunction, with 81.73% of test examples being correctly classified. We provide some discussion and analysis of the problem of conjunction in named entities, and we show that there are some cases which are ambiguous even for humans.

1 Introduction

Initially developed as a component task in information extraction (see, for example, (Grishman and Sundheim, 1996)), named entity recognition, whereby entities such as people, organizations and geographic locations are identified and tracked in texts, has become an important part of other natural language processing applications such as question answering, text summarisation and machine translation.

Named entity recognition consists of both identifying those strings in a text that correspond to named entities, and then classifying each such named entity string as being of a specific type, with typical categories being Company, Person and Location. Sometimes an additional step is introduced for coreference resolution, which establishes whether two named entities in a given document refer to the same (either physical or abstract) object; so, for example, we might determine that the named entity *International Business Machines Corporation* has the same real world referent as both *IBM* and *Big Blue*. We can go further and determine whether named entities in separate documents (hence, cross-document coreference resolution) refer to the same entities (see (Bagga, 2004)), although this is a much less explored area.

All of the above assumes that identifying individual named entities in text is a relatively straightforward and well-defined task. However, although there are reported high performance figures for named entity identification and classification in general, there are some categories of named entities that remain problematic. We will refer to those strings in a text that may correspond to named entities as **candidate named entity strings**; one type of candidate named entity string that is problematic, from a surface linguistic perspective, is the category that contains conjunctions. Consider the string *Australia and New Zealand Banking Group Limited*: in the absence of an appropriate domain lexicon, an occurrence of this string within a document could be interpreted as either being the name of one company, or as being a conjunction of a location and a company name.[1] Determining the correct interpretation is clearly important for any application which relies on named entity extraction.

We have been working with a data set from the Australian Stock Exchange (ASX). This data set consists of a large set of company announcements: for a variety of regulatory reasons, listed companies provide around 100000 documents to the ASX each year, and the ASX subsequently makes these available to users via the web. Our goal is to take this data set (and similar data sets) and to add value to the documents by making use of language technologies.

[1] Such a conjunction may appear pragmatically odd, but, as we argue below, rejecting possibilities on such grounds requires a complicated set of rules.

Proceedings of the Australasian Language Technology Workshop 2005, pages 7–14,
Sydney, Australia, December 2005.

The overall approach taken in this work is described in (Dale et al., 2004).

The significance of the kinds of ambiguities we introduced above depends, of course, on the extent to which the phenomenon of conjunctions in named entities is widespread. Our current work focuses on a subcorpus of 13000 ASX documents. From this subcorpus, we selected 45 documents at random; in these documents, there were a total of 545 candidate named entity strings, of which 31 contained conjunctions. This informal sampling suggests that conjunctions appear, on average, in around 5.7% of candidate named entity strings;[2] however, in some documents in our sample, the frequency is as high as 23%. These frequencies are sufficient to suggest that the seeking of an appropriate means of handling conjunctions is a worthwhile and important pursuit.

For present purposes, we define a candidate named entity string as any sequence of words beginning with initial capitals, with the possible inclusion of a single instance of the word *and* or the form *&* internal to the string.[3] An examination of the candidate named entity strings appearing in our corpus reveals four distinct uses of the conjunction, as exemplified in the following:

1. Oil and Gas Ltd
2. Agfa and Fuji
3. John and Mary Smith
4. Company Secretary Resignation and Appointment

In example (1), we have a single named entity that happens to contain an internal conjunction; in example (2), we have a conjunction of two distinct named entities; and in examples (3) and (4), we have conjunctions that, from a linguistic perspective, contain a form of ellipsis, so that one conjunct is incomplete on its own, but can be completed using information provided in the other conjunct.

That conjunctions are problematic has been noted before, in particular by Mikheev et al. (1998), who suggested the strategy of examining the preceding document context to identify candidate conjuncts that should be considered as separate named entities. Mikheev et al. mention this approach being part of their system used in the MUC-7 competition, but no data is reported on the accuracy of this kind of heuristic; in our experience, there are many cases where there are no antecedent mentions that can be used in this way. Furthermore, in the MUC-7 data, strings like *John and Mary Smith* were considered as one named entity, whereas we take the view that, for many information extraction applications, it is important to recognize that this string represents two distinct entities.

2 Problem definition

Following from the above, we distinguish four[4] categories of candidate named entity strings containing conjunctions.

A: Name Internal Conjunction: This category covers those cases where the candidate named entity string contains one named entity, where the conjunction is part of the name. Some examples from our corpus:[5] *Copper Mines and Metals Limited*, *Herbert P Cooper & Son*, *Ernst & Young*, *Acceptance and Transfer Form*, and *Fixing and Planning Phase*.

B: Name External Conjunction: This category covers those cases where the conjunction serves to separate two distinct named entities. Some examples from our corpus: *Proxy Form and Explanatory Memorandum*, *Hardware & Operating Systems*, *John Travolta and Robin Wright Penn*, and *EchoStar and News Corporation*.

C: Right-Copy Separator: This category of conjunction separates two named entities, where the first is incomplete in itself but can be completed by copying information from the right-hand conjunct. This is perhaps most common in conjunctions of

[2] For comparison, a check on the MUC-7 evaluation data shows that, in that corpus, the proportion of candidate named entity strings containing conjunctions is 4.5%, so our corpus appears not particularly unusual in this regard.

[3] This is clearly an overly restrictive definition, but it appears to account for a large proportion of the cases we are interested in.

[4] Conceptually, we might view the last two categories as subtypes of the more general category **Copying Separator**; however, it makes sense to keep the two categories separate, since the process of reconstructing the unelided conjuncts is different in each case.

[5] In what follows, we treat the ampersand (*&*) and the full lexical item *and* as being equivalent; however, as we discuss later, there are cases where it may be useful to distinguish the two forms.

proper names, as in *William and Alma Ford*, but appears in other contexts as well. Some examples from our corpus: *Connell and Bent Streets*, *Eastern and Western Australia*, and *Melbourne and Harvard Universities*.

D: Left-Copy Separator: This is similar to the previous category, but instead of copying information from the right-hand conjunct, in order to complete the constituent named entities we need to copy information from the left conjunct. Examples in our corpus: *Hospital Equipment & Systems*, *J H Blair Company Secretary & Corporate Counsel*.

In the data we have analysed, most examples are either Name Internal or Name External, and Left-Copy Separators are the rarest. It should be noted that the Copy Separator categories have been explored within linguistic treatments of conjunction, particularly as found in Categorial Grammar (see, for example, (Steedman, 1985)), although linguistic analyses tend to focus on conjunctions involving common nouns rather than proper names.

We could attempt to distinguish the different uses of the conjunction by means of some heuristics. For example, if a candidate named entity string matches the pattern ⟨GivenName and GivenName FamilyName⟩, the conjunction is probably a Right-Copy Separator; and if it matches the pattern ⟨CompanyName and CompanyName⟩, the conjunction is most likely a Name External Conjunction. However, analysis of a reasonably large sample makes it clear that there are many different cases to be considered, and the heuristics required are difficult to derive by hand; a significant reason for this is that the names of people, companies, and locations, as well as other less common named entity types, may occur in many different combinations.

Consequently, we decided to view the problem as one of classification: given a particular instance of the conjunction and its left and right conjuncts, we want to determine, via machine learning, which category the conjunction belongs to.

3 Experimental Setup

We carried out an experiment to determine how machine learning algorithms cope with the problem of conjunction classification in named entities. In the work described in this paper, we limited the experiment to candidate named entity strings containing a single occurrence of the conjunction *&* or *and*.

In our approach, before we attempt to disambiguate the conjunctions we first tag the constituents of each candidate named entity string with their types. This step also recognizes multi-word elements where there is no ambiguity (for example, in the case of unambiguous person and company names); for example, *Australia and New Zealand Banking Group Limited* should be recognized as a sequence of tokens whose types are marked as Loc and Loc Org CompDesig, where the second Loc tag corresponds to the pair of tokens *New Zealand*.

Table 1 lists the 21 tags we use to annotate the tokens. Some of these, such as Loc, Org, GivenName, AlphaNum, Dir, and PersDesig, are the same as those used by many other named entity recognizers; there are also two tags that come from part-of-speech tagging (Noun and Adj). In our corpus, we note that there are a number of frequently occurring key terms which can be thought of as closed class items that are highly indicative of named entity type, and so we also use a number of tags to annotate specific words that perform a key role in distinguishing the categories (Form, Son, Project, System, and Phase). In our corpus there are 160 occurrences of *Form(s)* tokens, 102 of *Son(s)*, 75 of *Project(s)*, 56 of *System(s)* and 4 occurrences of *Phase(s)*.

Some additional comments are appropriate by way of explanation of some of the tags:

- Fac (Facility) is a general-purpose category intended to cover 'domain objects': names of buildings, meeting places, and worksites. Examples are *Ashmore Tavern*, *Imperial Hotel*, *Parkway Plaza*, and *Solano Mall*.
- CompDesig (Company Designator) is used for those tokens that unambiguously mark the occurrence of a company name, such as *Ltd*, *Limited*, *Pty Ltd*, *GmbH*, and *plc*; we also use this tag for much longer and not so obvious multi-word sequences like *Investments Pty Ltd*, *Management Pty Ltd*, *Corporate Pty Ltd*, *Associates Pty Ltd*, *Family Trust*, *Co Limited*, *Partners*, *Partners Limited*, *Capital Limited*, and *Capital Pty Ltd*.

By assigning tags to tokens we obtain a **pattern** which represents the named entity candidate string. For example, for the string *Herbert P Cooper and Son Ltd* the

No	Tag	Meaning
1	Loc	The name of a location
2	Org	The name of an organization
3	GivenName	A person's given name
4	FamilyName	A person's family name
5	Fac	A facility
6	Initial	An initial in the range A-Z
7	CompPos	A position within a company
8	Abbrev	Abbreviation
9	PersDesig	A person designator
10	CompDesig	A company designator
11	Son	*Son(s)*
12	Dir	A compass direction
13	AlphaNum	An alphanumeric expression
14	Day	The name of a day
15	Month	The name of a month
16	Adj	An adjective
17	Noun	A noun
18	Project	*Project(s)*
19	System	*System(s)*
20	Phase	*Phase(s)*
21	Form	*Form(s)*

Table 1: The tags used for text annotation.

pattern is ⟨GivenName Initial FamilyName & Son CompDesig⟩.

For the purposes of machine learning, we then encode each pattern in the following way. We create an attribute for each of the 21 tag types for each of the left and right sides of a conjunction, for a total of 42 attributes. The attributes are of integer type with values $\{0, 1\}$, thus signaling either the presence or absence of a token of that type anywhere within either conjunct. We will refer to this encoding as the **Basic Encoding**.

On the basis of the results we obtained using the Basic Encoding (see Section 5.1), we created an additional five attributes for each conjunct: GivenNameCount, FamilyNameCount, InitialCount, NounCount and AdjCount. They serve as counts of the number of occurrences of the relevant token types, and have non-negative integer values. This extended set of 52 features will be referred to as the **Extended Encoding**.

With each data instance there is associated a ConjType attribute with the values {A,B,C,D}; this is used to encode the category of the conjunction in the training or test example.

The corpus used for our research consisted of a 13460 document sub-corpus drawn from a larger corpus of company announcements from the Australian Stock Exchange. The documents range in length from 8 to 1000 lines of text.

Choosing training and test examples was carried out in a number of steps. First, candidate named entity strings containing sequences of words with initial capitals, and an embedded conjunction, were extracted using a Perl script. This provided over 10560 candidate named entity string instances, corresponding to 6645 unique forms. For this experiment we did not collect candidate named entity strings containing lowercased prepositions and determiners such as *of*, *in*, *a*, and *the*, although clearly many relevant named entities will contain these elements.

From this set we randomly selected examples for our training and test data sets. Each random selection was followed by hand elimination of examples that turned out to be wrongly identified as candidate named entity strings in the text.[6]

The experiment consisted of two test runs, one using the basic encoding and the second using the extended encoding, as described above. In each case we used the same set of 348 training instances, and the same set of test data with 197 previously unseen examples.

Table 2 presents for each data set the distribution of examples across the four categories of conjunction.[7]

Data Set	A	B	C	D	Sum
Training	135	160	35	18	348
Test	55	119	15	8	197

Table 2: Data sets sizes and example distributions in categories.

4 The Algorithms

The experiment was conducted using the WEKA toolkit (Witten and Frank, 2005). This provides implementations of several machine learning algorithms, along with the data structures and code needed to perform data input

[6]Wrong identification occurred due to typographic features such as ASCII formatted tables, paragraphs in all upper case, sentences or titles where every word contained an initial capital, and punctuation errors in the source texts.

[7]By way of comparison, the corpus used for the MUC-7 final evaluation contains 53 strings corresponding to our category A and 124 strings for category B; the data from the training phase of the competition contains 28 category A strings and 81 category B strings. Recall from above that the MUC data does not recognize our categories C and D.

and output, data filtering and results evaluations and presentation.

After some initial exploration using a variety of algorithms for supervised machine learning available in WEKA, we chose six which gave the best results: the Multilayer Perceptron, two lazy algorithms (IBk and K*), and three tree algorithms: Random Tree, Logistic Model Trees and J4.8. We also include here the results for Naïve Bayes, given the popularity of this method in the field.

5 Results

We observe (see Table 2) that conjunctions of category B (Name External Conjunction) are the most frequent in our annotated data set. This gives us a simple baseline for comparison: by choosing the most frequent category by default, we would achieve a correct classification rate of 60.41%.

Tables 3 and 4 present detailed results of the two test runs using the classifiers trained on our feature set; recall that the extended encoding takes account of the number of instances of specific token types, rather than just a binary distinction between presence and absence. We show the number of correctly classified examples for both the training and test data sets.

Algorithm	Training	348	Test	197
Naïve Bayes	71.84%	250	68.53%	135
Mult. Perc.	91.95%	320	80.20%	158
IBk	92.24%	321	74.62%	147
K*	92.24%	321	74.62%	147
Random Tree	92.24%	321	77.16%	152
LMT	92.24%	321	81.22%	160
J4.8	87.36%	304	77.67%	153

Table 3: Results for basic encoding.

Algorithm	Training	348	Test	197
Naïve Bayes	70.11%	244	64.47%	127
Mult. Perc.	93.97%	327	77.67%	153
IBk	93.97%	327	75.13%	148
K*	93.97%	327	75.13%	148
Random Tree	93.97%	327	72.08%	142
LMT	93.68%	326	81.73%	161
J4.8	87.36%	304	77.67%	153

Table 4: Results for extended encoding.

5.1 Basic Encoding

When looking at the results from the classifiers, it turns out that patterns like ⟨Noun & Noun⟩ or ⟨Noun & Noun Noun⟩ made the biggest contribution to misclassification; also patterns built from combinations of several Nouns and some other tags such as Adj or Org had a significant impact on making the results worse. An interesting subgroup for us are patterns with tags Project, System and Form. It turns out that these domain specific tags were not sufficient to categorize test instances correctly.

The third large group of difficult examples are those that are represented by long patterns that consist of several kinds of tags. They also usually contain up to three Noun tags; examples of these patterns are as ⟨Noun GivenName Org & Noun Noun Noun⟩ and
⟨Abbrev CompDesig & Adj Noun CompDesig⟩.

The fourth group of problematic cases are patterns based on the FamilyName tag:
⟨FamilyName & FamilyName⟩,
⟨FamilyName & FamilyName Loc⟩,
⟨FamilyName Loc & FamilyName Loc⟩,
⟨FamilyName & FamilyName Noun⟩,
⟨FamilyName Noun & FamilyName Noun⟩,
⟨Initial & Initial FamilyName⟩.

5.2 Extended Encoding

An observation we made as a consequence of the results obtained for the basic encoding was that complexity and length of conjuncts appeared to play a role in misclassification; accordingly, for the extended encoding we introduced ten new attributes, in the form of five counters for each side of a conjunction. It turned out that new information was only slightly helpful for K* and LMT (in both cases the gain in performance was one example); for two algorithms there was no impact; and the Multilayer Perceptron and Random Tree algorithms encountered a significant drop in performance, by, respectively, five and ten examples (see Table 4). For both the Perceptron and Random Tree algorithms there were some cases where classification improved, and some for which it got worse.

For the Random Tree algorithm, nine additional examples were classified correctly; as expected, these were cases involving long patterns of up to 9 tags. However, 19 examples that were correctly classified using the basic encoding were misclassified with the extended encoding; most of these consisted of tags related to people or company names.

For the Multilayered Perceptron algorithm, the extended encoding improved performance on short patterns but made things worse on long

patterns.

5.3 General remarks

All the algorithms presented here performed far above the baseline in each test run. The results for the training data from each test (Tables 3 and 4) show that, in the training data, there are about 7% of examples which could be perceived as genuinely ambiguous. On investigation, the vast majority of these examples belong to several (usually two) categories and the most ambiguous are of ⟨Noun & Noun⟩ and ⟨FamilyName & FamilyName⟩ patterns, which we discuss later. The instances of these patterns appeared ambiguous to a human annotator.

The best result, 81.7259% of examples classified correctly, was achieved with the LMT algorithm and the extended set of attributes. The precision, recall and F-measure for this case are presented in Table 5. Table 6 provides a confusion matrix with the desired and actual classification of examples.

The LMT algorithm was also the best algorithm in the test run using the basic encoding of 43 features. The second best algorithm was the Multilayer Perceptron, which scored 80.2% and 78.2% with the basic and extended encodings respectively. However, the difference between the results for the two algorithms is not statistically significant.

Category	Precision	Recall	F-Measure
A	0.658	0.909	0.763
B	0.969	0.790	0.870
C	0.667	0.667	0.667
D	0.778	0.875	0.824
weighted mean	0.851	0.817	0.823

Table 5: Detailed accuracy by category of conjunction for best result (LMT, Extended Encoding).

A	B	C	D	→ classified as ↓
50	22	3	1	A
1	94	2	0	B
4	1	10	0	C
0	2	0	7	D

Table 6: Confusion matrix for best result (LMT, Extended Encoding).

6 Analysis

6.1 Data preparation

It is important to stress that our experiment was conducted in the specific domain of company announcements from the Australian Stock Exchange; these documents have some features that are not necessarily typical for others. In particular, texts in this domain frequently have some of the characteristics of legal documents, where many sometimes apparently arbitrary elements are given initial capitals: typical examples from our corpus would be expressions like *Primary and Secondary*, *Profit & Loss*, *Receivers and Managers* or *Resource and Reserve*. Many of these are high frequency terms, and so could be filtered out in a separate preprocessing stage; however, a complicating factor here is that casing is not used consistently by some authors. A more general problem is that of titles, for example titles of books, documents and document elements such as tables and headings; the unrestricted productivity of these kinds of terms means that they are not easily characterisable by patterns of the kind explored here, and would be more appropriately handled by a more syntactically driven model.

The results of our conjunction disambiguation process are very dependent on the tags assigned in the preprocessing stage. It can make a big difference, for example, whether a substring is recognized as a sequence of Nouns or as one Location name. Extensive gazetteers can play a role here, but some cases are ambiguous even for humans. For example, *Trustees Executors* is a company name;[8] but if this is not detected during tagging, it is tagged as ⟨N N⟩, which is much more ambiguous and impacts on performance. Similarly, a string like *Boyer and Haro Fields* can be difficult to analyze correctly without recourse to extensive world knowledge.

6.2 Error analysis

6.2.1 One Pattern, Many Categories

When investigating the misclassified strings we found that there are many cases where a given pattern belongs to more than one category. For example, we have the following in the training data:
String: *Ernst and Young Consulting*
Pattern: ⟨FamilyName & FamilyName Noun, A⟩
String: *National Parks and Wildlife Service*
Pattern: ⟨Adj Noun & Noun Noun, A⟩

[8] *Trustees Executors* was the first trustee company in New Zealand, established in 1881.

but in the test data we have:
String: *Boyer and Haro Fields*
Pattern: ⟨FamilyName & FamilyName Noun, C⟩
String: *General Meeting and Proxy Votes*
Pattern: ⟨Adj Noun & Noun Noun, B⟩

The ⟨Noun & Noun⟩ and ⟨FamilyName & FamilyName⟩ patterns are particularly prone to this ambiguity; we discuss these cases further below.

These examples suggest that our feature set is not capturing enough distinctions to enable correct classification; providing richer information about the candidate string would help, drawing from morphological, syntactic, semantic and even pragmatic features, should these be available.[9]

In different models created on the basis of training data by individual classifiers, a different category is assigned to these ambiguous patterns. Since the number of examples of a given pattern is not the same for particular categories, the number of misclassified examples differs for particular models. For example, there are ten instances of ⟨Noun & Noun⟩ pattern in the test data. Four of them are of category A, and six of them are of category B. The LMT algorithm assigns for this pattern probabilities $p_A = 0.455$, $p_B = 0.409$, $p_C = 0.136$ and $p_D = 0$. When a ⟨Noun & Noun⟩ example from the test data is to be classified, the classifier chooses category A as the most likely. On the other hand, the Multilayered Perceptron algorithm derives the following probabilities: $p_A = 0.414$, $p_B = 0.476$, $p_C = 0.110$ and $p_D = 0$. Consequently, it classifies all strings matching the ⟨Noun & Noun⟩ pattern as being of category B.

6.2.2 ⟨Noun & Noun⟩

As noted in Section 5, some of the most ambiguous candidate strings are those whose patterns are based on nouns and adjectives. This is not surprising, since these are very general tags capturing everything which is not recognized as a genuine proper name, an alphanumeric sequence of characters, or any of the specific distinguished terms captured by our other tags. In these cases the correct categorisation of the conjunction usually depends on the context, which sometimes can be even the whole text. General knowledge about the world is sometimes also essential; consider, for example, the strings *Gummy & Kipper* or *Showtime and Encore*, whose nature is quite unclear without recourse to other knowledge sources. This problem concerns mainly categories A and B, since initcapped general terms can appear equally freely in both cases.

In the test with the extended encoding, six out of 36 incorrect classifications involved the pattern ⟨Noun & Noun⟩; in each case, these should have been categorised as instances of category B (Name External Conjunction), but were classified as category A (Name Internal Conjunction). There were also another three examples of slightly more complex patterns containing more nouns:

String: *Tourism and Hotel Management*
Pattern: ⟨Noun & Noun Noun⟩
String: *Placement Shares and Options*
Pattern: ⟨Noun Noun & Noun⟩
String: *Country Comfort and Chifley*
Pattern: ⟨Noun Noun & Noun⟩

6.2.3 ⟨FamilyName & FamilyName⟩

The ⟨FamilyName & FamilyName⟩ pattern also causes problems in classification. Because many company names are created as a conjunction of two surnames, it happens that our training data contains more examples of this pattern for category A (Name Internal) than for category B (Name External), and so the model built by LMT considers category A to have probability of 0.704. However, there are still some cases when ⟨FamilyName & FamilyName⟩ does not denote an organization. So, regardless of the quality of the training data, it is worth checking whether somewhere in the text there exists a corresponding string with the pattern ⟨FamilyName & FamilyName CompDesig⟩.

6.2.4 Domain Dependent Substrings

As for any domain, it may be possible to identify a set of specific recurring strings or patterns that could be recognized in a preprocessing step, so that the learning algorithm does not need to deal with these cases. In our data, these turn out to belong to the set of strings that cause a range of problems. Two such examples are *Explanatory Notes and Proxy Form*, *Information Memorandum and Proxy Form*. In our corpus of candidate named entities, the string *Proxy Form* occurs 35 times. A reasonable strategy would be to assume that any string of the form *... and Proxy Form* involves the use of a Name External conjunction.

[9] In the example shown here, *Ernst and Young* could also be detected during preprocessing on the basis of a company name gazetteer, but such lists will never be complete.

7 Conclusions and Future Work

We have analyzed the problem of conjunctions in candidate named entity strings; we distinguished four categories of conjunction that appear in these strings, noted that the appropriate disambiguation of these is a problem that requires attention, and defined the problem as one of classification. We then conducted an experiment whose aim was to determine whether the problem could be solved by means of machine learning algorithms.

We have shown that there are instances of conjunction which are difficult even for humans to classify correctly. Very often the decision requires extensive analysis of the content, and the use of general world knowledge. Given the existence of such cases, the results demonstrated here with machine-learned classifiers are very encouraging. There are several regards in which the work reported here can be improved further.

1. We have restricted ourselves to candidate named entity strings which contain a single conjunction; however, there are of course cases where multiple conjunctions appear. One category consists of examples like *Audited Balance Sheet and Profit and Loss Account*, where again the kinds of syntactic ambiguity involved would suggest a more syntactically-driven approach would be worth consideration. Another category consists of candidate named entity strings that contain commas as well as lexicalised conjunctions.
2. In the work described here, we did not distinguish between the two variants of the lexicalised conjunction (i.e., *&* and *and*). Obviously, these two forms are not used in text completely interchangeably: for example, it is relatively unusual to separate two person names (as in *Alex and Bill Smith*) using an ampersand.
3. Candidate named entity strings can contain other closed class items, such as *of* and *the*; extension of the treatment here to this larger class of candidate strings will introduce new complexities, but at the same time these terms may provide useful disambiguating features.

An issue that may be restricted to corpora that have a broadly legal character is the frequent appearance of candidate named entities that are made up of common nouns. In contrast to a 'substring recognition' approach that relies on information from gazetteers, many of these cases might be amenable to a more syntactically sophisticated analysis, and this is one place where work on conjunction from a linguistic perspective might provide some leverage; however, there then remains the issue of determining which approach to use in a given case.
Named entities containing conjunctions, and named entities separated by conjunctions, constitute a form of ambiguity that needs to be handled for high accuracy named entity extraction. We have shown that machine learning can achieve good results in resolving these ambiguities.

8 Acknowledgements

The work reported here was carried out while the first author was a visiting scholar at the Centre for Language Technology at Macquarie University.

References

Amit Bagga. 2004. Cross-Document Coreference: Methodologies, Evaluations, and Applications. In António Branco, Tony McEnery, and Ruslan Mitkov, editors, *Proceedings of 5th Discourse Anaphora and Anaphor Resolution Colloquium, Portugal, 23-24 Sep., 2004.*

R. Dale, R. Calvo, and M. Tilbrook. 2004. Key Element Summarisation: Extracting Information from Company Announcements. In *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, 7th-10th December 2004, Cairns, Queensland, Australia.*

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference-6: A Brief History. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, Los Altos, Ca. Morgan Kaufmann.

A. Mikheev, C. Grover, and M. Moens. 1998. Description of the LTG System Used for MUC-7. In *Seventh Message Understanding Conference (MUC-7): Proc. of a Conf. held in Fairfax, Virginia, 29 April-1 May, 1998.*

M. Steedman. 1985. Dependency and Coordination in the Grammar of Dutch and English. *Language*, 61:523–568.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2nd edition.