Named Entity Extraction with Conjunction Disambiguation

Paweł Mazur* **and Robert Dale†**

Centre for Language Technology
Macquarie University, NSW 2109, Sydney, Australia
*mpawel@ics.mq.edu.au,
†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 extraction 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 some experiments using machine-learned classifiers to disambiguate the different uses of the conjunction, with 85% of test examples being correctly classified.

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.

Although there are reported high performance figures for named entity recognition and classification in general[1] there are some categories of named entities that remain problematic. One such category, from a surface linguistic perspective, is that of **candidate named entity strings** that contain 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. Determining the correct interpretation is clearly important for any application which relies on named entity extraction. Mikheev et al (Mikheev et al., 1998) suggest the strategy of examining the preceding document context to identify candidate conjuncts, but in many cases there are no antecedent mentions that can be used in this way.

The significance of these kinds of ambiguities depends, of course, on the extent to which the phenomenon of conjunctions in named entities is widespread. From our 13000-document Australian Stock Exchange corpus, 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; 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.

[1]See proceedings of MUC-6, MUC-7, CoNLL-2002 and CoNLL-2003 conferences

2. Conjunctions in Named Entities

We distinguish four categories of candidate named entity strings containing conjunctions.[2]

A: Name Internal Conjunction: This category covers those cases where the candidate named entity string contains one named entity and the conjunction is part of the name. Some examples from our corpus: *Dreamhaven Bedding & Furniture Limited*, *JB Were & Son*, *Farnell & Thomas*, *Acceptance and Transfer Form*.

B: Name External Conjunction: This category covers those cases where the conjunction serves to separate two distinct named entities. Some examples from our corpus: *Hardware & Operating Systems*, *Dean Pyle & Diana Helen Pyle*, 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 proper names, as in *William and Alma Ford*, but appears in other contexts as well. Some examples from our corpus: *Connell and Bent Streets*, *Central and Eastern Europe*, and *Vancouver and Toronto Stock Exchanges*.

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*.

[2]Conceptually, we might view the last two categories as subtypes of the more general category **Copying Separator**; however, in terms of processing, it makes sense to keep the two categories separate.

It should be noted that categories C and D 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 try 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⟩, it is probably of category C (Left-Copy Separator); and if it matches the pattern ⟨CompanyName and CompanyName⟩, it should be assigned to category B (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. Experiment

3.1. Experimental Setup

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.[3] Lowercased determiners (*the, a, an*) and preposition *of* were also allowed within these strings. This provided over 10924 candidate named entity strings.

Then training and test data sets were chosen from this set with the Perl `random()` function. We chose 400 examples for training and 200 examples for evaluation. Figure 1 presents the distribution of examples across the four categories of conjunction in both data sets.

Data Set	A	B	C	D	Sum
Training	120	243	16	21	400
Test	62	118	13	7	200

Figure 1: Example distribution in data sets.

Table 2 lists all the 19 tags we used to annotate the tokens. Some of these, such as Loc, Org, and GivenName, are the same as used by traditional named entity extractors; there are also some additional tags that we find useful, such as AlphaNum, Dir, and PersDesig. There are also two tags that come from part-of-speech tagging (Noun and Adj). Some additional comments are appropriate by way of explanation of some of the tags:

- Fac (Facility) is intended to cover names of products, buildings, meeting places, worksites, and similar.
- PersDesig (Person Designator) is used to annotate tokens such as *Mr*, *Mrs*, *Ms*, *Miss*, *Dr*, *Prof*, *Sir*, *Madam*, *Messrs*, and *Jnr*.
- 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*.
- The CompPos (Position) tag is used to mark the positions people can occupy within organizations, such as *Director*, *Secretary*, *Manager*, *Counsel*, *Managing Director*, *Member*, *Chairman*, *Chief Executive*, *Chief Executive Officer*, and *CEO*, and also for some bodies within organizations, such as *Board* and *Committee*.

No	Tag	Meaning
1	Loc	A name of a location
2	Org	A 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	A name of a day
15	Month	A name of a month
16	Adj	An adjective
17	Noun	A noun
18	Of	Preposition *of*
19	Deter	Determiners *the, a, an*

Figure 2: The tags used for text annotation.

Since we are using machined learned classifiers we encode patterns of training and test data. The encoding is done by creating an attribute for each of the 19 tag types for each of the left and right sides of a conjunction. The attributes are of binary type, thus signalling either the presence or absence of a token of that type on either side.

We also encode with one binary attribute the type of the conjunction, so the ampersand (*&*) and the lexical item *and* are distinguished.

Finally, we use one attribute with values {A,B,C,D} for encoding the category of the conjunction, for a total of 40 attributes per instance.

[3] We chose only those candidate named entity strings containing a single occurence of the conjunction *&* or *and*.

We ran two tests. In the first one we used 39 attributes and we did not distinguish the type of the conjunction. In the second test run we used the full encoding. This was to check how the conjunction separator type can be used to distinguish the conjunction category.

3.2. 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 and output, data filtering and results evaluations and presentation.

On the basis of some initial experiments, we chose the following classifiers: the Multilayer Perceptron and two tree algorithms, Random Tree and Logistic Model Trees (Landwehr et al., 2003).

There is a significant difference in the time of execution of particular algorithms. While the Multilayered Perceptron requires over two minutes to build the model, the Random Tree method does it in no more than half of second. LMT falls between these, with the time equal to about eighty seconds.

Random Tree is a method for constructing a tree that considers K random features at each tree node. It performs no pruning of the tree. In our experiment we set $K = 1$.

3.3. Results

Conjunctions of category B (Name External Conjunction) were the most frequent in our annotated test data set. This gives us a baseline for comparison: by choosing the most frequent category as the default one, we would achieve a correct classification rate of 59%.

All of the classifiers performed well above this baseline. Figure 3 presents detailed results for the first test run. We provide the number of correctly classified examples both for evaluations made on training and test data sets.

Algorithm	Training	400	Test	200
LMT	89.25%	357	80.0%	160
Mult. Perceptron	92.25%	369	82.0%	164
Random Tree	92.25%	369	83.5%	167

Figure 3: Results for the first test run.

Algorithm	Training	400	Test	200
LMT	90.00%	360	78.5%	157
Mult. Perceptron	93.25%	373	84%	168
Random Tree	93.75%	375	85.0%	170

Figure 4: Results for the second test run.

The best result in both test runs was obtained with the use of Random Tree method. This algorithm was also the quickest one.

Introducing information about the conjunction separator type turned out to be helpful for two algorithms.

Figure 6 shows the confusion matrix for the best results. We can see that the biggest contribution to overall misclassification was examples of category A which were classified as category B (ten instances). Both for category C and D there was only one misclassified test example.

Category	Precision	Recall	F-Measure
A	0.797	0.887	0.84
B	0.891	0.898	0.895
C	1.000	0.538	0.700
D	0.400	0.286	0.333
weighted mean	0.852	0.850	0.845

Figure 5: Detailed accuracy by category of conjunction for best result.

A	B	C	D	→ classified as ↓
55	12	1	1	A
5	106	4	4	B
0	0	7	0	C
2	0	1	2	D

Figure 6: Confusion matrix for best result.

3.4. Error Analysis

There were eighteen examples that were misclassified by all three algorithms in the first test run.

3.4.1. Noun-based patterns

Fourteen examples had patterns based on the Noun tag. We can distinguish two subgroups, equal in the number of examples. The first subgroup are patterns containing only Noun tags, like ⟨Noun & Noun Noun⟩. The second subgroup are patterns built on Noun tags but containing one additional tag.[4] In our experiment these were the Of, Org, Abbrev and Adj tags.

All of these examples were of a category other than B, while the models built by the algorithms assigned this category to examples consisting of many Nouns.

3.4.2. The ⟨FamilyName & FamilyName⟩ pattern

One would expect that examples of the pattern ⟨FamilyName & FamilyName⟩ are usually of the category B (Name External Conjunction). However, in our domain these are actually category A (Name Internal Conjunction) in most cases. This is because it happens quite often that a company name is made by conjunction of two family names.

In our test data set there was only one example of this pattern that did not express a company name and should be therefore classified as category B. However, it was wrongly classified as being of category A.

3.4.3. Long and complicated patterns

Long examples often consist of a large number of different tag types. For some of these cases, determining the proper category can be difficult even for humans. In some cases the solution may lie in a more sophisticated annotation, but in some cases, such as better annotating, product names, titles or unusual company names, there do not appear to be any obvious rules that can be determined on the basis of the

[4] To be precise, one of these examples contained actually two tags other than Noun: Adj and Abb.

tag types. In these cases, we require recourse to semantic and/or pragmatic knowledge based on the context of use.

4. Processing Model

In the standard model for named entity extraction there is a software module which takes a document for input, performs some operations on this document, and the output is a document with information about the occurences of named entities and their categories.

The internal architecture of this component can be very different from one system to another. It can be organized as a set of regural expressions written in a formal language using some general language processing platform, like SProUT (described, for example, in (Drożdżyński et al., 2004)) or GATE (see description in (Cunningham et al., 2002)). Another common approach is using machine learning algorithms in order to make the module less domain and language dependant. This approach was used in all systems developed for two Conferences on Computational Natural Language Learning (CoNLL-2002 and CoNLL-2003) – for detailed description of these systems see the references in (Sang, 2002; Sang and Meulder, 2003)). Finally, one can consider hybrid approaches where techniques and methods from both camps are drawn.

In our processing model, in order to reduce the number of extraction errors arising from candidate name entity strings containing conjunctions, we introduce two steps before the standard processing module for identifying named entities: **Component Named Entity Recognition** (CNER) and **Conjunction Disambiguation in Named Entities** (CDNE). The task of CNER is to assign preliminary tags to substrings of candidate name entity strings on both sides of the concjunction. These tags are essential for the CDNE step. The CNER component can be implemented using any of the approaches used in conventional NER work; in particular, gazetteers are particularly useful here.

In some cases it may happen that already at the stage of Component Named Entity Recognition we can decide on the type of the entire candidate name entity string. Consider the string *Ernst and Young*: if we find this in a companies' names list, this string can be classified at this step, and skipped by the subsequent CDNE step.

Conjunction Disambiguation in Named Entities is then implemented as a classification task which, based on the pattern of the candidate name entity string's parts, decides on the category of the conjunction. We have shown that this can be successfully implemented as machine learned classifier.

5. 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.

The results demonstrated here with machine-learned classifiers are very encouraging. We have also shown that utilizing the information about conjunction separator type can improve the results of classification.

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.

Acknowledgements The work reported in this paper was carried out while the first author was a visiting scholar at the Centre for Language Technology at Macquarie University. The second author acknowledges the support of the Capital Markets Cooperative Research Centre in carrying out this work.

6. References

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2004. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz*, 1:17–23.

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.

N. Landwehr, M. Hall, and E. Frank. 2003. Logistic Model Trees. In N. Lavrac et al., editor, *Proceedings of Fourteenth European Conference on Machine Learning, LNCS 2837*, pages 241–252, Berlin. Springer-Verlag.

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*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the 7th Conference on Natural Language Learning*, pages 142–147. Edmonton, Canada.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In Dan Roth and Antal van den Bosch, editors, *Proceedings of the 6th Conference on Natural Language Learning*, pages 155–158. Taipei, Taiwan.

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.