The ConceptMapper Approach to Named Entity Recognition

Michael Tanenblatt, Anni Coden, Igor Sominsky
IBM T.J. Watson Research Center
19 Skyline Dr, Hawthorne, New York, USA
E-mail: mtan@us.ibm.com, anni@us.ibm.com, sominsky@us.ibm.com

Abstract

ConceptMapper is an open source tool we created for classifying mentions in an unstructured text document based on concept terminologies and yielding named entities as output. It is implemented as a UIMA[1] (Unstructured Information Management Architecture (IBM, 2004)) annotator, and concepts come from standardised or proprietary terminologies. ConceptMapper can be easily configured, for instance, to use different search strategies or syntactic concepts. In this paper we will describe ConceptMapper, its configuration parameters and their trade-offs, in terms of precision and recall in identifying concepts in a collection of clinical reports written in English. ConceptMapper is available from the Apache UIMA Sandbox, using the Apache Open Source license.

1. Introduction

The proliferation of unstructured textual data in electronic format led to the development of many Natural Language (NLP) tools aimed at extracting knowledge from such documents. A basic task in NLP is named entity recognition (NER). There are three basic types of NER systems: machine learning, rule-based and lookup-based. If a lexicon, or vocabulary, can easily be compiled, the lookup approach has the advantage that no manually annotated corpus or model is needed. Such an approach can prove more conducive to the task of mapping terms to a controlled and coded vocabulary (terminology) and therefore associating semantic meanings and other properties to the terms. Machine learning approaches can often prove advantageous, in particular when no or only partial terminologies are available, but do not facilitate mapping into a coded vocabulary. Like lookup-based systems, rule-based approaches do not need manually annotated corpora, but do require expert linguistic knowledge and may be difficult to maintain, as the rule system is likely to be quite complex. A combined approach – first looking up terms in a controlled vocabulary, followed by a model driven algorithm can lead to identifying the largest number of mentions, though potentially sacrificing the precision of this identification.

Here we focus on the problem of identifying named entities (NE) specified in a coded terminology from unstructured text. We describe a highly parameterised and efficient lookup algorithm which can detect disjoint or variant multiword phrases. It is to handle these kinds of tasks that we created ConceptMapper. Accuracy results showing very high precision and recall are presented for the medical domain.

2. Related Work

ConceptMapper is being used for finding named entities in applications in the medical field, as well as with unstructured text documents in a variety of other domains, ranging from news to scholarly articles. Formal performance evaluations were performed in the medical domain. Though ConceptMapper is not limited to use in the medical domain, other domains were not evaluated due to the lack of an appropriate set of gold standard corpora, test corpora and terminologies.

In the medical domain, MetaMap released by the National Library of Medicine is a widely used system. This tool finds concepts specified in the UMLS terminology in unstructured text. Over the years, MetaMap was evaluated against multiple corpora (e.g. medical literature, sets of sentences and clinical notes) annotated with different subsets of the UMLS. Since the annotated clinical corpora are not available for comparative analysis—primarily due to de-identification requirements—we could not perform a head-to-head comparison against MetaMap. We do claim, though, that ConceptMapper should perform relatively equivalently to MetaMap, with mostly similar functionality, but without the limitation of being tied solely to UMLS. Furthermore, ConceptMapper is light weight, easily customizable both in terms of performance and in terminologies, and is released into Open Source.

There are a variety of tools in the literature[2] to address general named entity recognition. In the general domain, the focus is on identifying concepts such as people, places, organizations, addresses; in other words concepts which are semantically defined and not by a standardized and coded terminology. In the medical domain, the focus of evaluations of named entity recognition systems is against publicly available corpora, for instance the GENIA corpus, focusing on genes, proteins and there interactions. Evaluations against proprietary annotated clinical corpora can be found for instance in (Schuler, et al. 2008) which presented F-scores between 0.56 and 0.76 for an algorithmic named entity system. Again, direct comparisons cannot be executed.

[1] http://incubator.apache.org/uima/

[2] See http://site.cicling.org/2009/RCS-41/047-058.pdf for a comparison

3. What ConceptMapper Is

ConceptMapper is a highly configurable, flexible and accurate dictionary lookup tool, implemented as an open source UIMA component, as part of an NLP system. Only a tokenizer is required to have been run prior to ConceptMapper, though a sentence detector is also usually useful.

ConceptMapper was designed as a flexible tool to provide accurate mappings of unstructured text into named entities, as specified by controlled vocabularies in the form of dictionaries. Any properties associated with items mapped from the controlled vocabulary may also be associated with the NE's. Individual entries in a dictionary could consist of multiple tokens, and multi-token entries could potentially be matched against noncontiguous text, optionally in an order differing from way the tokens appear in the dictionary. ConceptMapper performs fast, and has been easily able to provide real-time results with multimillion entry dictionaries.

Lookups are token-based, and are limited to a specific context, usually a sentence, but can be configured for any context needed, such as a noun phrase or other NLP-based concepts.

All aspects of ConceptMapper's functionality can be configured:

1. The mappings from dictionary entries to resultant annotations: what type of annotations are created and what features are associated with those annotations.
2. The way input document tokens are processed
3. The choice of lookup strategy

Additionally, there are a set of post-processing filters, as well as an interface to create new filters. This allows for over-generating results during the lookup phase (explained below), then reducing the result set according to particular rules.

3.1 Dictionaries

The ConceptMapper dictionary is an implementation of the targeted terminology. The requirements on the design of the ConceptMapper dictionary were that it be easily extensible and that arbitrary attributes could be associated with all variants of an entry, but be overridden for any individual variant or even be unique to a variant. Additionally, the set of attributes could not be fixed, but customizable for any particular application.

The structure of a ConceptMapper dictionary is quite flexible and is expressed using XML (see figure 1). Specifically, it consists of a set of entries, specified by the token XML tag, each containing one or more variants (synonyms), the text of which is specified using by the base attribute of the variant XML tag. Entries can have any number of additional associated attributes, as needed. Individual variants inherit all attributes specified by their parent token (i.e., the canonical form), but can override any or all of them, or add additional attributes.

In the following sample dictionary entry, there are 6 variants, and according to the rules described earlier, each inherits the all attributes from the canonical form (canonical, CodeType, CodeValue, SemanticClass and POS), though the variants "colonic" and "colic" override the value of the POS (part of speech) attribute:

```
<token canonical="colon, nos"
       CodeType="ICDO" CodeValue="C18.9"
       SemanticClass="Site" POS="NN">
  <variant base=”colon, nos”/>
  <variant base=”colon”/>
  <variant base="colonic" POS="JJ" />
  <variant base="colic" POS="JJ" />
  <variant base="large intestine" />
  <variant base="large bowel" />
</token>
```

Figure 1. Sample Dictionary Entry

Since the results of running ConceptMapper are UIMA annotations, a method is provided to specify the mappings of attributes from the dictionary entries to the features of the resultant UIMA annotations.

The entire dictionary is loaded into memory, which, in conjunction with an efficient data structure, provides very fast lookups. Dictionaries with millions of entries have been used without any performance issues. The obvious drawback to storing the dictionary in memory is that large dictionaries require large amounts of memory; this is partially mitigated by the fact that the dictionary is implemented as a UIMA shared resource. This means that multiple annotators, such as multiple instances of ConceptMapper that are set up using different parameters, can all access the same instance of the dictionary, therefore loading it only once.

3.2 Tokenization

Since ConceptMapper matches tokens in text against tokens in dictionaries, it was designed specifically to allow the use of the same tokenizer for both the dictionary and for subsequent text processing, preventing missed matches due to different tokenization. Any tokenizer implemented as a UIMA annotator can be used. As an example of why this is important, consider the text:

poorly-differentiated/undifferentiated

which could be tokenized as 1, 3, or 5 tokens, as shown in figure 2:

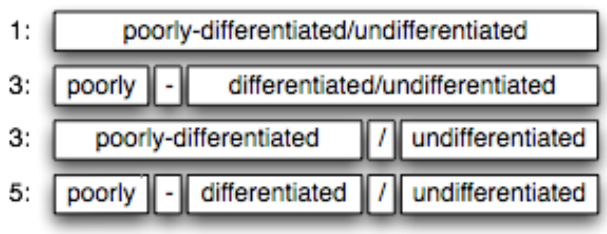

Figure 2: Tokenizations

Using the same tokenizer for dictionary entries and input documents prevents situations where a particular dictionary entry is not found, though it exists, because it was tokenized differently than the text being processed.

3.3 Input Document Token Processing

Input documents are processed on a token-by-token basis, one span (e.g., sentence or noun phrase) at a time. Hence it is always assumed that a tokenization module and the user-defined span creation module were applied prior to running ConceptMapper.

Some ways input token processing can be customized are:

1. Case sensitive or insensitive matching
2. Token text modifications: stemming, abbreviation expansions, spelling variants.
3. Use another feature of the token annotation entirely. This is useful for cases where spelling or case correction results need to be accessed instead of the token's original text.

One additional input control mechanism is the ability to skip tokens during lookups based on feature values of a particular feature or if the token's text is appears in a configurable stop-word list. Hence it is easy to skip, for example, all tokens with particular part of speech tag, or with some previously computed semantic class. For example, sometimes terms are interspersed within some multi-term dictionary phrase. For example:

infiltrating mammary **carcinoma**

In this case the dictionary specified "infiltrating carcinoma" as a diagnosis and "mammary" as an anatomical site, and each has an attribute reflecting that (i.e., "diagnosis" and "site", respectively). ConceptMapper can be configured to skip tokens by specifying the attribute/value set pair to skip in the "excludeList" parameter, (e.g. semantic class = "site"), so that if one pass over the text marked all the sites, a second pass could be configured to skip over all those with the site label.

Similarly, there is another configuration parameter that can be used instead to specify a set of feature values to use for inclusion. If supplied, both inclusion and exclusion sets are used to compute the tokens to include for lookup. The algorithm for selecting tokens to include during lookup is as follows:

```
if there is an includeList but no excludeList
    include annotation if feature value in includeList

else if there is an excludeList
    exclude annotation if feature value in excludeList

else
    include annotation
```

This provides a simple way to restrict the selection of pre-classified tokens, whether that pre-classification is done via previous instances of ConceptMapper or some altogether different annotator. For example, consider the sentence below, with the relevant feature's value in brackets following each token:

Oscar[*n*] Wilde[*n*] :[*p*] "[*p*] The[*q*] truth[*q*] is[*q*] rarely[*q*] pure[*q*] and[*q*] never[*q*] simple[*q*] .[*p*] "[*p*]

In this case, if the includeList contained a feature with the value "*q*", then the only tokens that would be considered during lookup would be:

The[*q*] truth[*q*] is[*q*] rarely[*q*] pure[*q*] and[*q*] never[*q*] simple[*q*]

The same tokens would be selected if there were no includeList, but the excludeList contained both "*n*" and "*p*", essentially excluding previously labeled punctuation and proper names.

3.4 Dictionary Lookup Strategies

The actual dictionary lookup algorithm is controlled by three user settable parameters. One specifies token-order independent lookup: for example, a dictionary entry that contained the variant:

```
<variant base='carcinoma, infiltrating'/>
```

would match against any permutation of its tokens. In this case, assuming that punctuation was ignored, it would match against both "infiltrating carcinoma" and "carcinoma infiltrating". Clearly, this particular setting must be used with care to prevent incorrect matches, but it does enable the use of a more compact dictionary, as all permutations of a particular entry do not need to be enumerated.

Another parameter that controls the dictionary lookup algorithm toggles between finding only the longest match vs. finding all possible matches. For the text:

… carcinoma, infiltrating ...

if there was a dictionary entry for "carcinoma" as well as the entry for "carcinoma, infiltrating", this parameter would control whether only the latter was annotated as a result or both would be annotated. Using the setting that indicates all possible matches should be found is useful when subsequent disambiguation is required.

The final parameter that controls the dictionary lookup algorithm specifies the search strategy, of which there are three. The default search strategy only considers contiguous tokens—not including tokens from the stop word list or otherwise skipped tokens (including punctuation), as described in the previous section—and then begins the subsequent search after the longest match. The second strategy allows for ignoring non-matching tokens, allowing for disjoint matches, so that a dictionary entry of:

A C

would match against the text:

A B C

This can be used as alternative method for finding "infiltrating carcinoma" over the text "infiltrating mammary carcinoma", as opposed to the method described above, wherein the token "mammary" had to have been somehow pre-marked with a feature and that feature listed as indicating the token should be skipped. On the other hand, this approach is less precise, potentially finding completely disjoint and unrelated tokens as a dictionary match. As with the default search strategy, the subsequent search begins after the longest match.

The final search strategy is identical to the previous, except that subsequent searches begin one token after the beginning of the previous match, instead of after the previous match. This enables overlapped matching. As with the setting that finds all matches instead of the longest match, using this setting is useful when subsequent disambiguation is required. As an example of overlapped matching, consider the medical text:

adenocarcinoma in polypoid tubulovillous adenoma

and a dictionary that contains both:

adenocarcinoma in polypoid adenoma

and:

adenocarcinoma in tubulovillous adenoma

the use of overlapped matching could be used to find both dictionaries over the given span of text.

3.5 Output Control

Given the fact that dictionary entries can have multiple variants, and that matches could contain non-contiguous sets of tokens, it can be useful to be able to know exactly what was matched. There are two parameters that can be used to provide this information. One allows the specification of a feature in the output annotation that will be set to the string containing the matched text. The other can be used to indicate a feature to be filled with the list of token annotations that were matched. Going back to the example in figure 2, where the token "mammary" was skipped, the matched string would be set to "infiltrating carcinoma" and the matched tokens would be set to the list of tokens "infiltrating" and "carcinoma".

Another output control parameter can be used to specify a feature of the resultant annotation to be set to contain the span annotation enclosing the matched token. Assuming, for example, that the spans being processed are sentences, this provides a convenient way to link the resultant annotation back to its enclosing sentence.

It is also possible to indicate dictionary attributes to store back into each of the original matched tokens. This provides the ability for the tokens themselves to be marked with information regarding what it was matched against. Going back to the example in figure 2, one way that the SemanticClass feature of the token "mammary" could have been labeled with the value "Site" was using this technique: a previous invocation of ConceptMapper had "mammary" as a dictionary entry, that entry had the SemanticClass feature with the value "Site", and SemanticClass was listed as an attribute to write back to the token as a feature. If, instead of "mammary" the match was against a multi-token entry, then each of the multiple tokens would have that feature set.

4. Parameter Configuration Comparison: A Case Study

We applied ConceptMapper to a named entity (NE) recognition task in the pathology domain, determining the trade-offs in performance for various combinations of parameter settings. The NE's we targeted were histological diagnoses (HD's) and anatomical sites (AS's), mapping them (respectively) to the morphology and topography entries of the ICD-O-3 (Fritz, et al, 2000). We varied parameters such as the search span, search strategy, and the use of stemming, running the tests against multiple dictionaries.

4.1 Dictionaries

We created separate dictionaries for HD's and AS's. While the dictionaries used ICD-O-3 as a starting point, we added additional synonyms according to a few rules: common abbreviations, adjectival forms, and commonly used shorthand expressions were added. In addition, we created alternative augmented versions, using synonyms from the SPECIALIST Lexicon[3] from the National Library of Medicine of the U.S. National Institutes of Health.

4.2 The Test

Precision, recall and F-measure for each of the 3 sets were computed over the test set of documents. The test corpus consisted of a "gold standard" set of 302 manually annotated (Coden, et al, 2009) colon cancer related English language pathology reports from Mayo Clinic. There were 976 HD's in the corpus, and 2316 AS's. We divided the corpus into 3 sets of 101, 101, and 100 documents. Sets 1 and 2 were used for development, and set 3 for evaluation of our algorithms. In the tests, we varied the following parameters:

- Stemming: on/off. The stemmer used was the Snowball stemmer[4]
- Dictionary with and without SPECIALIST Lexicon augmentation
- Matching limited to noun phrase or entire sentence
- Dictionary search strategy settings used:
- Order-independent matching (no negative effect for this domain)
- Find all matches: matching was not limited to longest match in dictionary.
- Allow for disjoint matches

[3] http://lexsrv3.nlm.nih.gov/SPECIALIST/Projects/lexicon/current/index.html

[4] http://snowball.tartarus.org/

These settings made post-processing necessary, but this was unavoidable due to the nature of the task: tokens could be referenced by multiple resultant terms. For example, the text:

Colon, rectum

was manually annotated with three annotations, one covering “Colon”, one covering “rectum” and another “Colon, rectum”. This interpretation is domain specific, reflecting the intended meaning of pathologists when writing these reports.

Tokenization and sentence boundary detection were performed using the IBM LanguageWare platform[5]. Noun phrases were identified using a proprietary shallow parser (Boguraev, 2000).

4.3 Results

The results of the tests are shown below in Table 1. The best results for the anatomical sites (AS) and histological diagnosis (HD) are shown in bold. As can be seen, both the base dictionary and the dictionary augmented using the SPECIALIST lexicon were tested. Table 2 describes how to interpret which settings were used in the tests.

		AS			HD		
	Test#	Prec’n	Recall	F-score	Prec’n	Recall	F-score
Aug’d	00	0.941	0.953	0.947	0.958	0.985	0.971
	01	0.902	0.838	0.869	0.855	0.910	0.882
	10	0.889	0.914	0.902	0.958	0.985	0.971
	11	0.871	0.816	0.843	0.855	0.911	0.882
Base	00	0.957	0.952	0.954	0.857	0.865	0.861
	01	0.918	0.837	0.876	0.779	0.815	0.797
	10	0.911	0.911	0.911	0.858	0.871	0.864
	11	0.890	0.815	0.851	0.781	0.822	0.801

Table 1: Test Results

Test #	**Stemmer**	**SpanFeatureStructure**
00	Off	SentenceAnnotation
01	Off	NPAnnotation
10	On	SentenceAnnotation
11	On	NPAnnotation

Table 2: Settings

where precision, recall, and F-score were computed by the standard definition, as follows:

	Entity in Gold Standard	Entity not in Gold Standard
Entity found by ConceptMapper	True Positive	False Positive
Entity not found by ConceptMapper	False Negative	True Negative

Table 3: Entity Labeling

Precision = numberOfTruePositives ÷ (numberOfTruePositives + numberOfFalsePositives)

Recall = numberOfTruePositives ÷ (numberOfTruePositive + numberOfFalseNegatives)

F-Score = 2 × Precision × Recall ÷ (Precision + Recall)

This compares quite well to the inter-annotator agreement of 95.7% (AS) and 97.8% (HD) over this corpus (Coden, et al 2009), as well as the state-of-the-art over a news article corpus from the MUC-7 competition[6].

It is clear that limiting lookups to noun phrases, as opposed to sentences, is undesirable. Examining the data, we could see that the main reason is that these target named entities often spanned beyond individual noun phrases. Limiting the lookup to noun phrases would make it impossible to identify the two diagnosis “intraductal carcinoma” and “invasive carcinoma” from the snippet “intraductal and invasive mammary carcinoma”. In addition, syntactic parsing is not 100% precise in ungrammatical and partial sentence style text as is common in the medical domain, increasing the error rate in the lookup due to non-identified noun-phrases.

A more surprising result was the impact of stemming. For histologies, the effect was negligible or nonexistent. This is easily explained by the fact that the dictionaries generally contained both plural and singular forms. For sites, the difference is pronounced. This is because the dictionaries are quite complete, so the use of stemming produces an increase in false positive matches.

Comparing the base vs. augmented dictionaries, a slight degradation in performance across the board appeared for anatomical sites. This is probably attributable to the breadth of coverage of the SPECIALIST lexicon, which introduced some spurious terms into the mix. On the whole, coverage was quite good, so those additional terms outweighed any potential gain. For histological diagnoses, on the other hand, the augmentation by the SPECIALIST lexicon had a profound positive effect. This can be explained by missing adjectival forms in the base dictionary which are provided by the SPECIALIST lexicon.

[5] http://www.ibm.com/software/globalization/topics/languageware/index.jsp

[6] http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/marsh_slides.pdf

5. Conclusions

We have introduced ConceptMapper, our open source UIMA-based NLP tool for mapping entries from a dictionary onto text documents. ConceptMapper is highly configurable, in terms of the way it processes input text, the way it performs search for the mapping, and the way it ultimately presents results.

Subsequently, we described a study wherein we compared varying a selected group of ConceptMapper's parameters over a corpus of colon cancer pathology reports for two different types of named entities: anatomical sites and histological diagnoses. The results show that very high precision and recall can be achieved using the ConceptMapper approach in this domain, enabling the direct mapping into a controlled vocabulary.

6. Acknowledgements

Our deepest thanks are extended to Dr. Eric W. Brown of IBM T. J. Watson Research Center for his work on the previous incarnation of this work, as well as our numerous colleagues at Mayo Clinic.

7. References

Ananiadou S, McNaught J, editors (2006). *Text Mining for Biology and Biomedicine*, Artech House

Boguraev, Branimir K. (2000). Towards finite-state analysis of lexical cohesion, *Proceedings of the 3rd International Conference on Finite-State Methods for NLP*, INTEX-3, Liege, Belgium.

Coden A., Pakhamov S.V., Ando R.K., Duffy P., Chute C.G., (2005). Domain-specific language models and lexicons for tagging, *Journal of Biomedical Informatics,* 12.

Coden, A.R., Savova, G.K., Buntrock, J.D., Sominsky, I.L., Ogren, P.V., Chute, C.G., de Groen, P.C., (2007). Text Analysis Integration into a Medical Information Retrieval System: Challenges Related to Word Sense Disambiguation, *MedInfo*, Brisbane, Australia.

Coden, A., Savova, G., Sominsky, I., Tanenblatt, M., Masanz, J., Schuler, K., Cooper, J., Guan, W., de Groen, P.C. (2009). Automatically extracting cancer disease characteristics from pathology reports into a Disease Knowledge Representation Model, *Journal of Biomedical Informatics*, 42, pp. 937-949

Friedman C, Johnson S.B., Starren J. (1995). Architectural requirements for multipurpose natural language processor in the clinical environment. *Proceedings of the Annual Symposium on Computer Applications in Medical Care*, pp. 347-51.

Fritz A., C. Percy, Jack, A., Shanmugaratnam, K., Sobin, L., Parkin, D.M., Whelan, S., Editors (2000). *International Classification of Diseases for Oncology*, Third Edition. World Health Organization

IBM (2004). Unstructured Information Management, *IBM Systems Journal* Vol 43, No. 3

Jimeno, A., Jimenez-Ruiz, E., Lee, V., Gaudan, S., Berlanga, Rebholz-Schuhmann, R.,D., Assessment of disease named entity recognition on a corpus of annotated sentences, *MBC Bioinformatics*, 9

Meystre S.M., Savova G.K., Kipper-Schuler K.C., Hurdle J.F. (2008). Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research, *IMIA Yearbook of Medical Informatics*

Poesio, M., Vieira, R., (1998). A corpus-based investigation of definite description use. *Computational linguistics*, 24(2), pp. 183-216

Schuler, K., Kaggal, V., Masanz, J., Ogren, P.V., Savova, G. (2008). System Evaluation on a Named Entity Corpus from Clinical Notes, *Proceedings of the Sixth International Language Resources and Evaluation.*