Related Entity Finding: University of Waterloo at TREC 2010 Entity Track

Olga Vechtomova
Department of Management Sciences
University of Waterloo
Waterloo, Ontario, Canada
ovechtom@uwaterloo.ca

1. Introduction

The University of Waterloo participated in the Related Entity Finding task of the Entity track. Our goal is to investigate whether related entity finding problem can be addressed by unsupervised approaches that rely primarily on statistical methods and common linguistic tools, such as named-entity taggers and syntactic parsers. We approach the related entity finding problem by first retrieving documents in response to the query, and extracting an initial set of candidate entities from the text of the documents. As a separate step, we automatically construct a set of seed entities, which represent hyponyms of the target entity category specified in the narrative, and then rank the candidate entities by their similarity to the seeds. An example of the target entity category name is "authors", extracted from the narrative "Authors awarded an Anthony Award at Bouchercon in 2007" (2009 topic #14). The system extracts category names from the free-text narrative, finds seed entities belonging to each category, and computes the similarity of candidate entities to the seeds.

2. Methodology

Figure 1 provides an overview of the main components of our system. In the first stage (rectangle 1 in Figure 1), the system retrieves an initial set of documents for the query from the Web. Only the sentences containing query terms plus one preceding/following sentence are retained. Named Entity tagging is applied to these sentences, and candidate entities are extracted and ranked. In the second stage, the target category name is automatically identified from the topic narrative, and in stage 3 the system finds hyponyms of this category name, and selects seed entities from the hyponyms. In stage 4, the entities (candidates and seeds) are represented as vectors of weighted grammatical dependency relations, and pairwise (candidate-seed) similarity is calculated. In stage 5, candidate entities are ranked by similarity to all seeds. Stage 1 is described in Section 2.1, while stages 2-5 are presented in Section 2.2.

2.1 Extracting candidate entities

As the first step, the queries to retrieve top documents from the Web are generated from the "entity name" and "narrative" sections of the Entity track topics according to the algorithm in Figure 2. The objective is to extract named entities and other noun phrases from the topic. For this purpose we use a Part-Of-Speech tagger (Brill, 1995), a Noun Phrase chunker (Ramshaw and Marcus, 1995), and a list of titles of Wikipedia pages. The resulting queries are then used to retrieve the top 50 documents from a Web search engine. Our motivation to use 50 is to keep the number of documents for subsequent in-depth analysis reasonably small, and at the same time have sufficient amount of text to extract entities from.

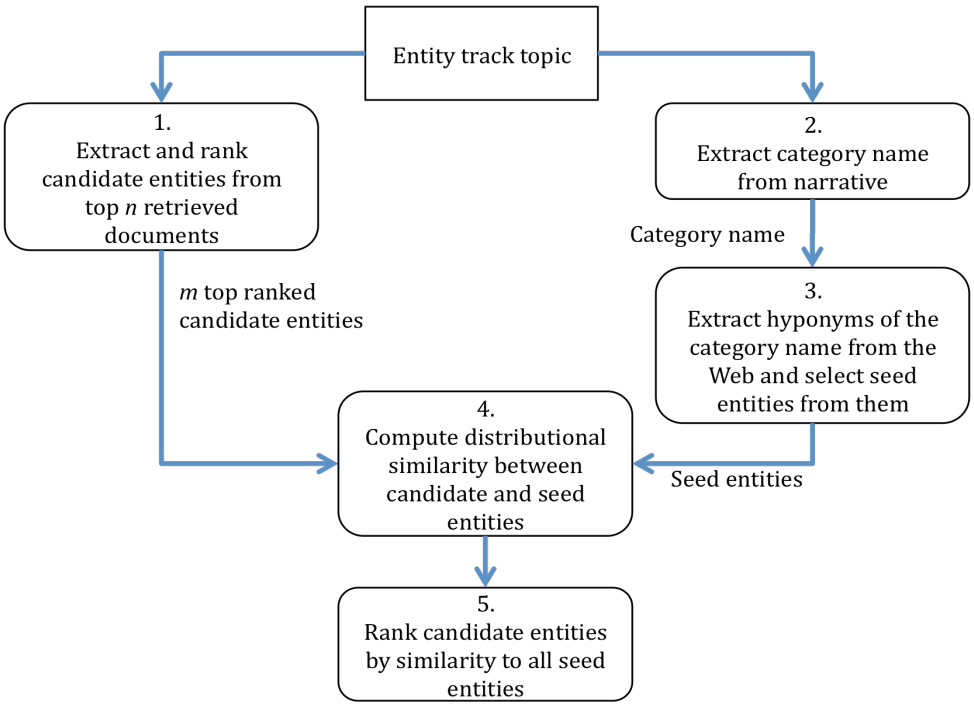

Figure 1. Components of the related entity finding system.

```
For each topic
  Process narrative using a POS tagger and Noun Phrase chunker.
  Get all possible contiguous ngrams from the "entity name" and
  "narrative" sections of the topic.
  Sort ngrams in the descending order of the number of words
  For each ngram n
    If n OR possessive form of n OR singular form of n matches a
    Wikipedia title
        m = form of n matching a Wikipedia title
    Elsif n is a unigram and does not match a Wikipedia title
        m = n
    If m is not a stopword
        Remove "the" or "a" if found at the beginning of m
        If m is extracted from narrative
            If m is part of any NP in the NP-chunked narrative
                If m has not been written to the query before
                    Add m to the query as a phrase (string in
                    quotation marks)
```

Figure 2. Algorithm for processing queries

The retrieved 50 documents are parsed to remove HTML tags, script and style sections, and broken into sentences. We then extract sentences that contain at least one query term. If a query term is a noun, the system attempts to match its singular and plural forms. For each such sentence, we also extract one sentence before and one after. Let this set of sentences be *{S}*. The sentences are then processed by the LBJ-based Named Entity Recognizer (Ratinov and Roth, 2009). The NER tagger only assigns the categories of "Location", "Organization", "Person" and "Miscellaneous". In the Entity track 2010, the target entity type specified in the topic is one of the following: "organization", "person", "location" and "product". For topics

belonging to the first three types, we extract all entities tagged with the corresponding labels. However, for topics of category "Product" we extract entities labeled as "Organization" and "Miscellaneous".

Having extracted candidate entities from the top 50 documents retrieved for each topic, we then rank them by *TF*IDF*, where *TF* is the frequency of the entity in the sentence set *{S}* extracted from the top 50 retrieved documents, and *IDF* is calculated by using the number of documents containing the entity in the ClueWeb09 Category B collection. Since the TF*IDF-ranked list can be quite large and may contain a lot of noise, we select the top 200 entities for the use in subsequent stages. In the following sections we will refer to this ranked list of 200 entities as "candidate entities".

2.2 Ranking candidate entities by the similarity to the target entity category

Since the chosen NER tagger can only be used to identify entities of a few broad categories, such as organisations and people, the list of candidate entities can be noisy. This is further compounded by the NER tagger errors. To refine the list of entities, we decided to investigate the applicability of the distributional similarity principle, which is based on the observation that semantically close words occur in similar contexts. If we know a small number of correct seed entities, we can rank the candidate entities by the distributional similarity to them. Many methods reported in the literature that utilised the distributional similarity principle are semi-supervised methods, such as (Lin, 1998). These start with the list of known seed words and then find other words that are distributionally similar, and therefore, likely to be semantically close to the seed words. The problem in the Related Entity Finding task is that the seed words are not given. However, the topic narratives have descriptions of the categories of entities that are to be retrieved. Our approach is to generate a list of seed entities based on the given entity category names. For example, the narrative of the 2009 training topic #3 is "Motorsport series that Bridgestone officially supports with tyres." We developed a method to extract the category name from the narrative, i.e. "motorsport series" in this topic, and adapted a method for the automatic acquisition of the hyponymy relation proposed by Hearst (1992) to find entities that belong to the category (in this case "motorsport series"), which we then use as seed entities. Furthermore, a new method was developed to compute the distributional similarity between seed entities and candidate entities, and rank the entities by similarity to all seed entities.

2.2.1 Extracting category names from topic narratives

To extract category names, the narratives are first processed using Brill's Part-of-Speech (POS) tagger (Brill, 1995) and a Noun-Phrase chunker (Ramshaw and Marcus, 1995). Then a set of rules is applied to select one of the initial noun phrases (NPs) from the narrative. Generally, the first noun phrase in the narrative is selected as the category name, unless it is a personal pronoun or the noun "searcher". Other rules include splitting a NP containing conjunction (e.g. "and", "/", "or") into two or more NPs (see example for 2009 topic #4 in Table 1). Table 1 lists examples of NP-chunked narratives of the training topics and the extracted category names.

Table 1. Examples of NP-chunked narratives and extracted category names (2009 training topics)

Topic	NP-chunked narrative	Extracted category name
4	[7.30/CD report/NN presenters/NNS and/CC reporters/NNS] ./.	7.30 report presenters 7.30 report reporters
5	[Members/NNS] of/IN [John/NNP Key's/NNP cabinet/NN] ./.	members
6	[What/WP] [products/NNS] did/VBD [Nokia/NNP] sell/VB throughout/IN [its/PRP$ history/NN] ,/, [that/WDT] are/VBP still/RB sold/VBN [today/NN] ?/.	products
7	[I'd/NNP] like/VB to/TO find/VB [which/WDT] [operating/VBG systems/NNS] [I/PRP] can/MD use/VB on/IN [the/DT EEE/NNP PC/NN]	operating systems
10	[Which/WDT fashion/NN labels/NNS] belong/VBP to/TO [the/DT LVMH/NNP group/NN] ?/.	fashion labels

2.2.2 Identifying seed entities

After the category name is extracted from the topic narrative, the next step is to find entities that belong to this category. We adapted the unsupervised hyponymy acquisition method proposed by Hearst (1992). Hearst's method consists of using six domain- and genre-independent lexico-syntactic templates that indicate a hyponymy relation. The templates and examples of sentences found for the 2009 training topic #7 are shown in Table 2.

Table 2. Hearst's patterns for hyponym acquisition

Template	Examples based on training topic #7
NP such as {NP,} {(or|and)} NP*	…on some multi-user operating systems such as Windows XP, Windows Vista, Mac OS X, OpenSUSE, Ubuntu and Fedora.
such NP as {NP,} {(or|and)} NP*	Experienced in such operating systems as MS-DOS, Windows, Windows NT/2K/XP/2K3, Solaris, Linux, FreeBSD.
NP {, NP}{,} or other NP*	If you want to run Windows, Linux, or other operating systems on…
NP {, NP}{,} and other NP*	PostScript font installation - Unix, Linux and other operating systems.
NP{,} including {NP ,} {or | and} NP*	We will cover each of the major operating systems, including DOS, Windows 9x/NT/2000/XP, and UNIX/Linux.
NP {,} especially {NP,} {or | and} NP*	…but the program is used mainly on other operating systems, especially Linux.

For each topic, six queries are constructed using the above templates and the category name is extracted from the topic narrative. For example, the query for the first template in Table 2 is: "operating systems such as". Each query is submitted to one of the commercial search engines as a phrase (i.e. quote-delimited). If the total number of pages retrieved by all six queries is fewer than 10, the first word in the category name is dropped and the search is repeated. If again it returned fewer than 10 pages, the first two words are dropped, and so on until either 10 or more pages are retrieved, or the remaining category name consists of only a single word, in which case we use whatever number of pages were found. If a category name is a single word, the query includes the topic title in addition to the template, in order to minimise the extraction of unrelated entities.

The documents retrieved for each query are processed to remove HTML tags, and split into sentences. The sentences containing the hyponymy lexico-syntactic patterns are then processed using the LBJ-based NER tagger (Ratinov and Roth, 2009). Depending on the expected position of hyponyms in the lexico-syntactic pattern, NEs either immediately preceding, or following the pattern are extracted. If several NEs are used conjunctively, i.e., separated by a comma, "and" or "or", all of them are extracted. For each topic, we extract only NEs with tags corresponding to the entity type specified in the topic, i.e. NEs tagged with "ORG", "PER", "LOC" and ("MISC"|"ORG") are extracted for topics with entity types "organization", "person" "location" and "product" respectively. Below is an example of the output of the NER tagger for topic #7:

"…on some modern multi-user operating systems such as [MISC Windows XP] , [MISC Windows Vista] , [ORG Mac OS X] , [PER OpenSUSE] , [ORG Ubuntu] and [ORG Fedora] ."

Since, the entity type for the 2009 training topic #7 is "product", the following entities are extracted as hyponyms of "operating system": "Windows XP", "Windows Vista", "Mac OS X", "Ubuntu" and "Fedora". In this sentence the tagger erroneously tagged "OpenSUSE" as person, therefore it is not extracted.

One problem with using all found hyponyms as seed entities is that they can be unrelated to the topic. Some category names extracted from topic narratives are very broad, such as "products". Applying the above

hyponym acquisition algorithm based on such high-level hypernym is bound to produce a very large number of topically-unrelated hyponyms. Computing distributional similarity between candidate entities and these hyponyms is likely to be ineffective and possibly detrimental to performance. We therefore must ensure that we use only those hyponyms as seeds, for which there exists some evidence of relationship to the topic. For this purpose, we defined as seeds the intersection of found hyponyms and entities extracted from the top 50 documents retrieved for the initial query as described in Section 2.1. For example, for topic #7, the following hyponyms were identified as seeds: "FreeBSD", "Mac", "Linux", "Ubuntu", "Windows XP", "Microsoft Windows", "Unix", "Microsoft". If only one seed word is identified as a result of this process, we do not perform entity re-ranking on this topic, and keep the original rank order, i.e. by TF*IDF. We believe that one seed entity is an insufficient representation of the target entity category, and may substantially degrade performance if it is incorrect.

2.2.3 Computing distributional similarity between candidate and seed entities.

Distributional similarity between entities is computed based on how similar their contexts of occurrence are in text. In their simplest form, contexts could be words extracted from windows around entity occurrences. Alternatively, they could be grammatical dependency relations, with which an entity occurs in text. The use of grammatical dependency relations is more constraining in calculating entity similarity, and allows us to identify entities belonging to the same semantic category. For example, grammatical dependency relations in Table 3 tend to occur with noun phrases referring to sports people.

Table 3. Examples of grammatical dependency relations.

Grammatical dependency relation	Source sentence
win *V:subj:N* Rubens Barrichello	**Rubens Barrichello won** the Italian Gran Prix.
teammate *N:appositive:N* Rubens Barrichello	His **teammate, Rubens Barrichello**, managed second with a 1:35.681 lap at the end of the session.

Since we are interested in identifying entities that are of the same semantic category as the seed words, we decided to use grammatical dependency relations for calculating entity similarity. For each seed entity we retrieve 200 documents from ClueWeb09 Category B using BM25 (Robertson et al., 1995) implemented in Wumpus[1]. Each document is split into sentences, and sentences containing the entity are parsed using the Minipar[2] syntactic parser (Lin, 1993) to extract grammatical dependency triples. Each dependency triple consists of two words/phrases and a grammatical relation that connects them (see examples in Table 3). These dependency triples are transformed into features representing the context of each entity. To transform a triple into a feature, we replace the entity name in the triple with 'X', e.g., "win V:subj:N Rubens Barrichello" is transformed into "win V:subj:N X". To avoid using features that are specific to only one or few seed entities, only features that co-occur with at least 50% of all seed entities are retained. For each seed entity we build a vector consisting of these features and frequencies of their co-occurrence with this entity in the 200 retrieved documents.

A vector for each of the 200 candidate entities obtained in Stage 1 (Section 2.1) is generated following the same procedure as above for seed entities. Each vector consists only of those features (and frequencies of their co-occurrence with the candidate entity) that co-occur with at least 50% of all seed entities.

To compute the similarity between the vectors of seeds and candidate entities, we use BM25 with query weights. For each seed and candidate entity we calculate a query adjusted combined weight, QACW (Spärck Jones et al., 2000). In the QACW formula, the vector of the seed entity is treated as the query and the vector of the candidate as the document:

[1] http://www.wumpus-search.org/

[2] http://webdocs.cs.ualberta.ca/~lindek/minipar.htm

$$QACW_{c,s} = \sum_{f=1}^{F} \frac{TF(k_1+1)}{K+TF} \cdot QTF \cdot IDF_f \qquad (1)$$

Where: F – the number of features that a candidate entity c and a seed entity s have in common; TF – frequency of feature f in the vector of candidate entity; QTF – frequency of feature f in the vector of the seed entity; $K = k_1*((1-b)+b*DL/AVDL)$; k_1 – feature frequency normalisation factor; b – document length normalisation factor; DL – number of features in the vector of the candidate entity; $AVDL$ – average number of features in all candidate entities.

In order to calculate the *IDF* of a feature, we need to have access to a large syntactically parsed corpus, such as the ClueWeb09 Category B collection. Since we do not have such a resource, and it is computationally demanding to produce one, we approximate *IDF* of a feature with the *IDF* of its component word. For example, for the feature "win V:subj:N X" we calculate the *IDF* of "win" by using its document frequency in the ClueWeb09 Category B collection.

Arguably, when calculating the similarity of candidate entities to seed entities, we should take into account how strongly each seed entity is associated with the original TREC topic. Candidate entities similar to those seed entities, which have weak association with the topic, should be downweighted compared to those candidate entities, which are similar to seed entities strongly associated with the topic. We quantify this association by using the *TF*IDF* weight of the seed entity calculated as described in Section 2.1. Thus, the candidate entity matching score with all seed entities is calculated according to:

$$EntitySeed_c = \sum_{s=1}^{S} TFIDF_s \times QACW_{c,s} \qquad (2)$$

Only the entities with an *EntitySeed* weight of greater than zero are retained. The final score for a candidate entity is calculated as a linear combination of the candidate entity's *TF*IDF* and *EntitySeed* weights, as defined in the following equation:

$$TFIDFEntitySeed_c = \beta \times TFIDF_c + (1-\beta) \times EntitySeed_c \qquad (3)$$

3. Results

The runs submitted are UWAT1, which uses the TF*IDF method (Section 2.1), and UWAT2, which uses the seed similarity method described in Section 2.2.

Since the requirement of the Related Entity Finding task in 2010 is to return one homepage for each retrieved entity, we developed a simple homepage finding algorithm, which consists of retrieving the top 10 webpages for each entity from a major commercial Web search engine, filtering out a small number of common URLs, such as "dictionary.com", "facebook.com", "linkedin.com", and using as a homepage the top ranked page that exists in the ClueWeb09 Category A collection. The results are summarised in Table 4, and the NDCG@R performance difference of UWAT2 from UWAT1 by topic is shown in Figure 3.

Table 4. Results on the 50 topics.

Run	nDCG@R	P10	MAP	RPrec	Rel. Retr.	Pri. Retr.
UWAT1 (TF*IDF)	0.1264	0.0957	0.0608	0.1033	95	151
UWAT2 (TFIDFEntitySeed)	0.1393** (10.2%)	0.1106 (15.6%)	0.0722* (18.8%)	0.1223 (18.4%)	96	154

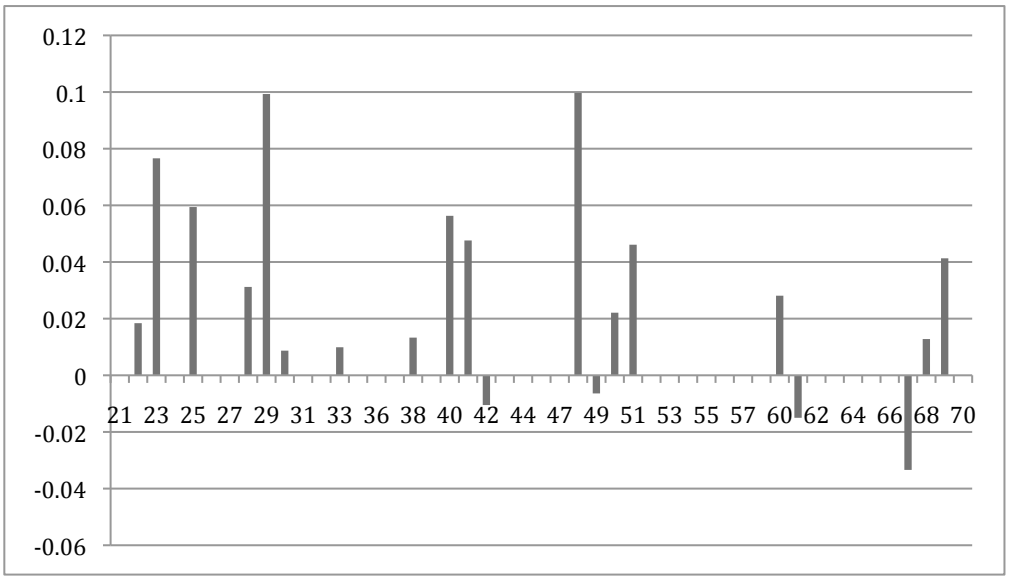

Figure 3. Difference of UWAT2 from UWAT1 in NDCG@R by topic.

As can be seen from Table 4, re-ranking and pruning candidate entities by calculating their similarity to seed entities improves performance significantly. Figure 3 demonstrates that most of the topics are improved by using the re-ranking process.

4. Conclusion

In this paper we describe our approach to the Related Entity Finding task of the Entity track of TREC 2010. The method relies primarily on statistical and linguistic methods. In our approach, candidate entities are extracted using a NER tagger from the documents retrieved in response to the query. As a separate step a set of seed entities is generated by finding hyponyms of the entity category name given in the narrative. The candidate entities are then compared to the seeds by the similarity of the grammatical dependency relations they co-occur with. Entities that have a higher similarity to a larger number of seeds are ranked higher in the final list. The results suggest that when we apply the entity re-ranking by similarity to seed entities, we get a statistically higher performance compared to when only the first stage of entity ranking (by TF*IDF) is performed. A possible future extension of this work is bootstrapping additional seeds from the initial set of seeds identified using the hyponymy method described here.

* and ** are statistically significant at 0.05 and 0.01 levels (2-tailed paired t-test).

References

Brill E. (1995) Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics, 21(4), pp. 543-565.

Hearst, M. A. (1992) Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th Conference on Computational Linguistics - Volume 2 (Nantes, France, August 23 - 28, 1992), 539-545.

Lin, D. (1993) Principle-Based Parsing Without OverGeneration. In Proceedings of ACL-93. pp. 112-120. Columbus, OH.

Lin, D. (1998) Automatic retrieval and clustering of similar words. In Proceedings of the 17th international Conference on Computational Linguistics - Volume 2 (Montreal, Quebec, Canada, August 10 - 14, 1998), 768-774.

Ramshaw L. and Marcus M. (1995) Text Chunking Using Transformation-Based Learning. Proceedings of the Third ACL Workshop on Very Large Corpora, MIT.

Ratinov L. and Roth D. (2009) Design Challenges and Misconceptions in Named Entity Recognition. In Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL).

Robertson S.E., Walker S., Jones S., Hankock-Beaulieu M., Gatford M. (1995) Okapi at TREC-3. In Harman D. (Ed.) Proceedings of the Third Text Retrieval Conference, NIST, Gaithersburg, U.S., pp.109-126.

Spärck Jones, K., Walker, S., & Robertson, S. E. (2000). A probabilistic model of information retrieval: Development and comparative experiments. Information Processing and Management, 36(6), 779–808 (Part 1); 809–840 (Part 2).