LIA at TAC KBP 2012 English Entity Linking track

Ludovic Bonnefoy
LIA - University of Avignon
ludovic.bonnefoy@etd.univ-avignon.fr

Patrice Bellot
LSIS - Aix-Marseille University
patrice.bellot@lsis.org

Abstract

This paper describes our participation in the English Entity Linking task at KBP 2012.

1 Introduction

For our first participation to the English Entity Linking task our goal was to exploit the source document of a topic to find more information and context about the source entity and thus to be more confident in our selection of the corresponding knowledge base (KB) entry. To do so we analyzed the source document to extract variants of the source entity via coreference chains and extracted relation tuples which contains one of the variants found as an argument. These new information are used to iteratively find new documents about the entity which are analyzed in the same fashion in order to consolidate and extend information retrieved. After a predetermined number of steps all the variants and relations were finally used to rank candidate KB entries.

This paper is broken down as follows : we first describe how documents were analyzed ie. how variants and relations were extracted and weighted from them, then Section 3 explains how we selected candidate entries from the KB, Section 4 gives the ranking parameters for each run submitted. Finally results are presented and analyzed.

2 Finding more evidences about an entity

The main step of our system is about extracting the maximum of information from the document collection which can be useful for candidate generation and ranking. At first, the source document was analyzed in order to extract such information, then documents from the collection were retrieved by querying a search engine. This search step was guided by using the set of extracted evidences through query expansion. Top ranked documents were selected and analyzed in the same fashion. This process was then repeated until a fixed number of loops was done.

2.1 Name variations resolution

The first kind of information we cared about was name variation of the source entity (and not only of the KB entries as many systems usually did). Entity name variations finding is a crucial step in order to avoid to miss a correct KB entry when genereating candidates for a difference in a the surface form of the name. This is even more true when the source entity is an acronym. Moreover having found name variations of the entity could improve candidate ranking.

2.1.1 Acronym expansion

In our approach we started to test if the source entity was an acronym. We naively considered the source entity as an acronym if all letters were uppercased. We looked for an extended version of it in the source document : we selected as an extended version of the acronym all word sequences with an uppercase at the beginning of each words and exactly matching the letters of the acronym.

2.1.2 Finding name variations

Documents were analyzed to extract the coreference chains (by means of the Stanford

CoreNLP tool [1]). A co-reference chain was considered as referring to the source entity if at least one the element in the chain was a named entity which contains at least a part of the source entity. If so, all the named entities in the co-reference chain were considered as name variations of the source entity. This recall-oriented approach select a lot of incorrect name variations (for instance for the source entity *"Hillary Clinton"* if a chain contains *"Bill Clinton"* our approach will select it as a name variation along with all entities in the same coreference chain).

2.1.3 Variants weighting

To avoid to give too much importance to named entities incorrectly selected as name variations we weighted each variants according to their frequencies and the confidence in documents from which they were extracted :

$$w(v_i) = \sum_d tf(v_i, d) * conf(d)$$

where w is the weight of the variant v_i, d is one the analyzed documents, $tf(v_i, d)$ is the term frequency of v_i in d and $conf(d)$ is a confidence score in d. If d is the source document then $conf(d) = 1$ else the confidence score is the score associated to d by a search engine when d was retrieved.

2.1.4 Types weighting

The type of the source entity is a valuable information for filter out candidate entries which do not match it, however it is not provide in the topics so we inferred with types associated to variants. The Stanford CoreNLP tool is able to deal with three broad types of named entities (person, location and organization) which correspond to the KB's entries types. The weight gave to each type for a topic was the sum of the weights of each variant of this type :

$$w(t_i) = \sum_{v \in V} w(v) \times isOfType(v, t_i)$$

where t_i is a type, $w(v)$ is the weight of one variant v and where isOfType() is equal to one if type t_i is associated to v, 0 else.

[1] http://nlp.stanford.edu/software/corenlp.shtml

2.2 Relations

Both source entity related entities and the relations themselves could be good criterion to choose between different candidate entries by penalizing those which did not shared them. For instance, the entity "Barack Obama" have the strong relation "president of" with USA and this information could appear in the corresponding KB entry in the text or facts. Using relations could also be a way to guide efficiently the selection of documents to analyzed and from which extract more variants and relations. To do so, for each document analyzed for variations resolution, relation extraction was performed. We choose to do not specified in advance relations of interest to be domain independent. It is why we used Reverb [2] which was specially designed to extract relations not known in advance. Reverb extract relations with a learned set of weighted rules. All relation containing a variant was selected. Relations were weighted the same way as variants were :

$$w(r_i) = \sum_d tf(r_i, d) * conf(d)$$

where w is the weight of the relation r_i (both the related entity and the relation itself), d is one the analyzed documents, $tf(r_i, d)$ is the term frequency of r_i in d and $conf(d)$ is a confidence score in d. If d is the source document then $conf(d) = 1$ else the confidence score is the score associated to d by a search engine when d was retrieved.

2.3 Source entity focused documents retrieval

Once the source document was analyzed we wanted to got new documents about the source entity in order to find new variants and relations or at least to validate those already found. The all document collection was first indexed by a search engine (Indri [3]). To be able to find documents focused on the source entity we relyied on the extracted variants and relations to guide the document retrieval. The search engine was asked to return documents containing at least one tuple $v_i r_j e_j$ (where v_i is one of the extracted variants and r_j and e_j a relation and the associated related entity). Elements of the query were weighted according to their importance W. The x

[2] http://reverb.cs.washington.edu

[3] http://www.lemurproject.org/indri/

top ranked documents were selected to be analyzed where x had to be picked to kept a good balance between precision (selected documents had to be about the source entity) and performances (processing time of a query) (in our experiments x was fixed to 3). After retrieved documents were processed this step was repeated a predetermined number of times.

3 Candidate Generation

Candidate generation is the action to look for all the KB entries which could correspond to the source entity. Entries are selected according to :

- Exact Match : entries whose title matches one the name variations were selected. Text in parentheses (eg. *"Bush (band)"*) or after a comma (eg. *"Bush, Cornwall"*) was ignored for this test.
- Link Match : we added a "aliases" field to each KB entry which contained all the alternative names for the entry found in fact links of the KB.

4 Candidate Ranking

Candidate entries were finally ranked.

First, titles of candidates were scored according to the valued variants collected :

$$s(t_i e) = \sum_{v_j \in V} w(v_j) \times (1 + w(type)) \times (1 + \log(tf(v_j, kb)) \quad (1)$$

where t_i is the title of a candidate entry for a source entity e, V is the set of variants and $w(v_j)$ is the weight of the j^{th} one, $w(type)$ is the weight of the candidate entry type for V and $tf(v_j, kb)$ is the number of times that v_j variant appears in the KB as text of a link to the candidate entry.

As many participants did **??** we used the cosine similarity between a vector representing the candidate entry and an other used to models the source entity context. Two different elements was considered for the candidate's vector :

- facts only
- both text and facts

In both cases a bag of words (unigrams) approach was adopted. To build the source entity's vector three elements was considered :

- the name variations
- the related entities (2.2)
- concatenated texts of a set of documents retrieved in the same fashion as presented in subsection 2.3

As for the candidate's vector, in both cases a bag of unigrams was used.

A nil id was assigned to topics for which no candidates were found. Two nil topics were assigned to a same cluster if their entity source match perfectly.

5 Runs

Our team submitted seven runs which correspond to different combinations of scores and information used :

wiki text used	r-t-t (LIA1)	r-e-t (LIA2)	e-t-t (LIA4)	v-t-t (LIA8)
without	r-t-f (LIA5)	r-e-f (LIA6)	r-n-n (LIA9)	

Table 1: Runs names

A run's name is composed as a-b-c where :

- a : gives information about what kind of information was used for the query expansion :
  - r : all name variations and relations (both related entities and the relations themselves) were used as explain in 2.3
  - e : name variations and related entities were used, relations were excluded
  - v : only name variations guided the document retrieval
- b : give details about how source entities' vectors were build for context similarity
  - t : the text of documents retrieved
  - e : the related entities
  - n : no similarity
- c : give details about how candidate entries' vectors were build

– t : text and facts of the entry
– f: facts only
– n : no similarity

For all runs, the source entity focused documents retrieval step was done five times and each times the 3 top ranked documents retrieved were used.

6 Results

	All	in KB	not in KB
r-t-t	.180	.223	.132
r-e-t	.171	.204	.132
e-t-t	.183	.220	.139
v-t-t	.164	.224	.094
r-t-f	.174	.211	.132
r-e-f	.171	.205	.132
r-n-n	.173	.208	.132

Table 2: Official results at TAC KBP 2012 with the $B^3 + F1$ metric

Official results are presented in Table2. Theses results are very low compared to results obtained by other participants. So it makes difficult for us to analyse our results, compare our different runs and learn from them because differences are not significant using the official metric. However some analysis can be made.

The two main answers are our recall oriented candidate generation and our nil clustering algorithm. In mean we return nil for only 300 topics (on the 1050 nil topics in the gold standard) and it affect dramatically our $B^3 + F1$ score. To test the importance to return nil when needed we did a baseline run which consist to return a unique nil id to each topic (all of them). This run obtains a score of .442 (see "AllNil-Unique" Table 3), far better than our runs and close to the median runs. This result shows that our recall strategy was not relevant and our system will greatly benefit from a decision algorithm to determine if or not a nil id should be returned for a topic given candidate entries scores and characteristics. Table 3 also presents an other run called "AllNil-clustered". This run was built in the same way than "AllNil-unique" but our clustering step was applyied. We can see that our clustering approach was to much naive and hurt the results (.250).

	$B^3 + F1$	recall	mrr
allNil-unique	.442		
allNil-clustered	.250		
noQE-noSim-noLinkedM	.165	.458	.383
noQE-noSim-withLinkedM	.183	.579	.569
e-t-t	.183	.758	.520

Table 3: Non officials evaluation with B^3+F1 for all topics, recall and mrr of corret entries after candidate generation and ranking steps

We saw what we think are the two main issues of our approach but with the official metric it is difficult to estimate the complete performances of our system. We made a third baseline system to evaluate how the query expansion process help to retrieve correct entries (improve the recall during the candidate generation step) and if using name variations found in the KB improve the ranking. This run named "noQE-noSim-noLinkedM" do not rely on the query expansion step and links in the KB are ignored. If we compare it to "noQE-noSim-withLinkedM" - for which links in KB were used - we can see that using links greatly improve both recall and ranking. Now if we compare e-t-t to the latter we can see that the recall is dramatically increased by the query expansion process but the ranking is a little bit lower. These last result allows us to say that this is not the candidate generation step which does not perform well in our system but the ranking which is not sufficient (the harmonic rank of the correct answer is 1.92).

7 Conclusions

We presented our KBP English Entity Linking system for our first participation in the task in 2012. Our approach was based on a query expansion and pseudo-relevance feedback process used to find entity source's name variations and find more occurrences of it in order to know more about its context of appearance. This process was iterative and all information founds in previous steps were used to guided the next one.

Our system obtained poor results mainly because of our recall oriented approach, a too much naive approach for nil clustering and the absence of a threshold or a decision algorithm to decide to return nil or

not when candidate entries may be not relevant.

For this first participation we decided to do not rely on past data to learn and optimize some parameters, however those data are available and we think of using them to rank candidates, deciding or not to return a nil id and if yes to cluster them in our further experiments.