Stanford-UBC Entity Linking at TAC-KBP

Angel X. Chang,[†] Valentin I. Spitkovsky,[†] Eric Yeh,[†]
Eneko Agirre,[‡] Christopher D. Manning[†]

[†] Computer Science Department, Stanford University, Stanford, CA, USA

[‡] IXA NLP Group, University of the Basque Country, Donostia, Basque Country

Abstract

This paper describes the joint Stanford-UBC knowledge base population system for the entity linking task. We modified our 2009 approach, which was based on frequencies of Wikipedia back-links, providing a context-independent mapping from strings to possible Wikipedia titles. We then built on this foundation, taking into account named-entity recognition (NER) and coreference resolution information to disambiguate entities within a document. Simple heuristics — both context-independent and context-sensitive — were sufficient for our runs to score higher than the median entry. This year an additional challenge was organized, in which systems could not use the free text from the Wikipedia pages associated with the knowledge base nodes. Our context-independent run outperforms all systems that participated, but unfortunately we did not submit to this subtrack.

1 Introduction

The 2010 Text Analysis Conference (TAC) Knowledge Base Population (KBP) track includes two tasks: entity linking and slot filling. We participated in the first task with a system similar to the one we prepared for 2009 TAC-KBP [ACJ$^+$09] — a robust combination of several techniques inspired by the word sense disambiguation (WSD) literature [AE06]. For 2010 TAC-KBP, we submitted two runs: one based on a much simplified system, relying mainly on the dictionary we developed last year; and a second approach that uses a simple heuristic for incorporating context into entity disambiguation.

2 The Dictionary

Our first task was to construct a rudimentary static mapping from strings to relevant Wikipedia articles. We scored each association using the number of times that the string appeared as the anchor-text of an article's incoming hyperlinks. Note that such dictionaries can disambiguate any of the dictionary's keys directly, simply by returning a highest-scoring article. To construct the dictionaries, we augmented the official KB with all English pages from the March 6th, 2009 Wikipedia dump and all English Wikipedia pages in a Google crawl; from each, to the extent possible, we extracted Wikipedia titles, URLs and redirects.

2.1 Remapping

To collect titles and URLs that in fact refer to the same article, we first remapped all entries not in the official KB to URLs using Wikipedia's canonicalization algorithm. We then connected any two URLs that appeared together either in an official Wikipedia redirect or that were redirected at crawl-time. For each title and URL, we then extracted a connected component (which could be quite large). From each component, we chose a single representative, with preference given first to members of the official KB, then to non-redirect pages from the Wikipedia dump, followed by redirect pages, and finally pages that did not appear in the KB or the Wikipedia dump. Within each preference category, we resolved ties lexicographically. Here is an example of a typical cluster merged to be represented by the bolded entry:

Route_102_(Virginia_pre-1933)	State_Route_102_(Virginia_1928)
State_Route_102_(Virginia_1928-1933)	State_Route_102_(Virginia_pre-1933)
State_Route_63_(Virginia_1933)	State_Route_63_(Virginia_1933-1946)
State_Route_63_(Virginia_1940)	State_Route_63_(Virginia_pre-1946)
State_Route_758_(Lee_County,_Virginia)	**Virginia_State_Route_758_(Lee_County)**

All of the entries above are taken to refer to the bolded URL. In what follows, we use the single representative from each cluster.

2.2 Core Dictionary

The core dictionary maps arbitrary strings to sorted lists of (remapped) Wikipedia URLs and associated scores. There are five possible scores, and each string-URL pair must at least have one: booleans $\{\mathtt{c}, \mathtt{d}, \mathtt{t}\}$ and rationals $\{\mathtt{w}, \mathtt{W}\}$. The presence of booleans in a mapping from `<str>` to `<url>` indicates that:

c) *clarification* — either `<str>` or `<str> (...)` is the title of some page in the cluster represented by `<url>`;
d) *disambiguation* — a disambiguation page titled `<str>` links to some page in the cluster represented by `<url>`;
t) *title* — a page titled `<str>` is in the cluster represented by `<url>`.

The rational `w`:x/y, $x \neq 0 < y$ indicates that of the y inter-Wikipedia links with anchor-text `<str>`, x pointed to a page represented by `<url>`; similarly, `W`:u/v, $u \neq 0 < v$, indicates that of the v external links into Wikipedia with anchor-text `<str>`, u pointed to a page represented by `<url>`.

The list of URLs for each string is sorted in decreasing order by $(x + u)/(y + v)$, taking $x = 0$ when `w` is absent, $u = 0$ when `W` is absent, and $(x+u)/(y+v) \equiv 0$ when both are absent, with ties resolved lexicographically. We refer to this dictionary as exact (EXCT), as it contains precisely the strings found using the methods outlined above. For example, for the string `Hank Williams`, it returns the following eight associations:

0.997642	Hank_Williams	W:936/938 c d t w:756/758
0.00117925	Your_Cheatin'_Heart	W:2/938
0.000589623	Hank_Williams_(Clickradio_CEO)	c w:1/758
0.000589623	Hank_Williams_(basketball)	c w:1/758
0	Hank_Williams,_Jr.	d
0	Hank_Williams_(disambiguation)	c
0	Hank_Williams_First_Nation	d
0	Hank_Williams_III	d

2.3 Fuzzy Flavors

In addition to exact lookups in the core dictionary, we entertain two less strict views. In both cases, an incoming string now maps to a *set* of keys, with their lists of scored URLs merged in the intuitive way: a boolean score is present in the merged score if it is present in any of the original scores for the URL; rational scores are merged using the formula

$$a/b + c/d + \cdots \rightarrow \frac{a + c + \cdots}{b + d + \cdots}.$$

- **LNRM**: We form the lower-cased normalized version $l(s)$ of a string s by canonicalizing its UTF-8 characters, eliminating diacritics, lower-casing the UTF-8 and throwing out all ASCII-range characters that are not alpha-numeric. If the resulting string $l(s)$ is empty, then s maps to no keys; otherwise, it maps to all keys k such that $l(s) = l(k)$, with the exception of $k = s$, to exclude the key already covered by EXCT. Thus, $l($`Hank Williams`$) =$ `hankwilliams` $= l($`HANK WILLIAMS`$) = l($`hank williams`$)$, etc. The combined contribution of all but the original string with this signature yields five suggestions (most of which overlap with EXCT's):

0.952381	Hank_Williams	W:20/21 c t
0.047619	I'm_So_Lonesome_I_Could_Cry	W:1/21
0	Hank_Williams_(Clickradio_CEO)	c
0	Hank_Williams_(basketball)	c
0	Hank_Williams_(disambiguation)	c

- **FUZZ**: We define a metric $d(s, s')$ to be the byte-level Levenshtein edit-distance between strings s and s'. If $l(s)$ is empty, then s maps to no keys once again; otherwise, it maps to all keys k, $l(k)$ not empty, that minimize $d(l(s), l(k)) > 0$. Note that this again excludes $k = s$, as well as any keys covered by LNRM. For `Hank Williams`, some strings whose signature is exactly one byte away include `Tank Williams`, `Hanks Williams`, `hankwilliam`, and so on. Together, they recommend just three articles (two of which were already covered by EXCT/LNRM):

0.631579	Tank_Williams	c t w:12/12
0.315789	Hank_Williams	W:6/7
0.0526316	Your_Cheatin'_Heart	W:1/7

2.4 Other Flavors

In addition to the core and fuzzy dictionaries, we experimented with a number of alternatives, two of which proved useful enough to be included in the final submissions.

- **GOOG**: Queries the Google search engine[1] using the directive `site:en.wikipedia.org` and the string `<str>`, keeping only URLs that start with `http://en.wikipedia.org/wiki/` and using inverse ranks for scores.

- **CAPS** (context-sensitive): Queries a repository of Wikipedia pages using Lucene.[2] In addition to the string `<str>`, it includes all non-trivially capitalized words in the document containing `<str>` for context via the (logically-redundant) construct `<str>` AND (`<str>` OR `<context>`), which boosts the articles containing *both* `<str>` and `<context>`, and using the scores returned by Lucene.

3 Context-Sensitive, Heuristic Disambiguation

Our main context-dependent approach attempts to disambiguate an entity mention by identifying other possible mentions of the same entity in text. The intuition here is that while some acronyms (such as "ABC") are ambiguous, there may be other mentions of the same entity in the document that uniquely identify it (e.g., "American Broadcasting Company"). To keep things simple, we assume that the longest entity mention is least ambiguous and use *that* in our dictionary lookup, instead of the original mention.

We find the set of other possible mentions for a given entity's string by first running the Stanford NER system [FGM05] on the document's text to find all entity names. For all occurrences of the target entity string, we determine whether it was part of a longer NER chunk. We then use the Stanford deterministic coreference system [RLR^{+}10] to find all entity names that are coreferent with the target entity. In addition, since our dictionary provides a list of Wikipedia articles to which the target mention could refer, we can identify all strings in the document matching any of their titles. We then use the longest string obtained by extending the current entity mention to an NER chunk, coreference resolution, and matching Wikipedia titles in the document, and use *that* as the new entity name to query the dictionary. This yields a new set of possible matching Wikipedia articles. To avoid noise (say, from a bad coreference suggestion), we restrict this set, intersecting it with the original list of possible Wikipedia titles, obtained earlier (by using the original mention to query the dictionary for possible articles). Finally, we select the top-ranked title from this set.

Below we give motivating examples for our proposed heuristics, taken from the 2009 TAC-KBP entity linking evaluation:

String	`Abbott` [60 choices in the LNRM dictionary]
Top Choice	`Abbott_Laboratories`
Correct Choice	`Bud_Abbott`
Context	`Comedian Bud Abbott of Abbott and Costello died in Woodland Hills, California.`
Heuristic	Expanding the target string, `Abbott`, to the named entity span containing it, `Bud Abbott`, leads us to the correct entity.

String	`ABC` [253 choices]
Top Choice	`American_Broadcasting_Company`
Correct Choice	`All_Basotho_Convention`
Context	`...say victory over the newly-formed All Basotho Convention ( ABC ) is far from certain...`
Heuristic	Using coreference resolution to match the target string, `ABC`, to `All Basotho Convention`, correctly determines the entity.

String	`ABC` [still 253 choices]
Top Choice	`American_Broadcasting_Company`
Correct Choice	`Australian_Broadcasting_Corporation`
Context	`...he told ABC...`
Earlier Sentence	`...he told the Australian Broadcasting Corporation...`
Heuristic	Looking for candidate Wikipedia titles that also match the target string, `ABC`, in the document, we find that the longest such title is `Australian Broadcasting Corporation` — that of the correct entity.

Notice that the last method — looking for matching Wikipedia titles in a document — works correctly for all three examples.

[1]`http://www.google.com/`

[2]`http://lucene.apache.org/`

4 Supervised Disambiguation

In addition to plain lookups, we applied machine learning techniques to do supervised disambiguation of entities. For each target string, we trained a multiclass classifier to distinguish the more relevant of the possible articles. Using the text surrounding the target string in the document, this classifier scored each possible entity among the mappings provided by the dictionary.

Inter-Wikipedia links and the surrounding snippets of text comprised our training data. When one article linked to another, we assumed that the anchor text was synonymous with the linked Wikipedia entity (as collapsed by the remapper, described above). Since the dictionary could return entities having no incoming Wikipedia links for the target string, we restricted the list of possible entities to those with actual links for the exact (raw, unnormalized) target string. Our snippets were spans of text, consisting of 100 tokens to the left and 100 to the right of a link, extracted from the October 8th, 2008 English Wikipedia dump.

Depending on the number of Wikipedia spans that contained the target string as linked text, we sometimes supplemented our training data with spans that linked to a possibly related entity:

- spans with exact matches (same anchor-text as the target string) if there were more than 1,000 such spans. For instance, since there were over 3,000 Wikipedia spans linking from the string `Pittsburgh`, we simply used just those Wikipedia spans for training;

- spans with all possible entities for the target string if there were fewer than 1,000 spans with exact string matches. For example, for `ASG`, there were only 13 Wikipedia spans linking from the exact string `ASG`, so we also used spans linking to `Abu Sayyaf`, `ASG (band)`, as well as other entities associated with the string `ASG` as training data.

From the spans, we extracted the following features for training:

- the anchor text;
- the lemmas in the span;
- lemma for noun/verb/adjective in a four-token window around the anchor text;
- lemma and word for noun/verb/adjective before and after the anchor text;
- word/lemma/part-of-speech(POS) bigram and trigrams around the anchor text.

For example, the Wikipedia article for `SuperFerry` has the following span of text that links the string `Abu Sayyaf` to the URL `Abu_Sayyaf`:

```
On February 27 , 2004 , SuperFerry 14 was bombed by the Abu Sayyaf terrorists killing 116 people .
It was considered as the worst terrorist attack in the Philippines .
```

We would use this span (actually a longer version, covering 100 tokens to the left and also to the right of `Abu Sayyaf`) as a training instance of `ASG` mapping to `Abu_Sayyaf` with the features shown on the next page.

anchor text		`unigram Abu_Sayyaf`
lemmas in the span		`win_cont_lem_context terrorist`
		`win_cont_lem_context kill`
		...
lemma for noun/verb/adjective in a 4 token window around the anchor text		`win_cont_lem_4w be`
		`win_cont_lem_4w bomb`
		`win_cont_lem_4w kill`
		`win_cont_lem_4w people`
		`win_cont_lem_4w terrorist`
lemma and word for noun/verb/adjective before the anchor text	noun (lemma)	`prev_N_lem SuperFerry`
	noun (word)	`prev_N_wf SuperFerry`
	verb (lemma)	`prev_V_lem bomb`
	verb (word)	`prev_V_wf bombed`
lemma and word for noun/verb/adjective after the anchor text	adjective (lemma)	`post_J_lem bad`
	adjective (word)	`post_J_wf worst`
	noun (lemma)	`post_N_lem terrorist`
	noun (word)	`post_N_wf terrorists`
	verb (lemma)	`post_V_lem kill`
	verb (word)	`post_V_wf killing`
bigrams around the anchor text	lemma before	`big_lem_func_+1 the Abu_Sayyaf`
	lemma after	`big_lem_cont_-1 Abu_Sayyaf terrorist`
	POS before	`big_pos_+1 DT J`
	POS after	`big_pos_-1 J N2`
	word before	`big_wf_func_+1 the Abu_Sayyaf`
	word after	`big_wf_cont_-1 Abu_Sayyaf terrorist`
trigrams around the anchor text	lemma before	`trig_lem_func_+1 by the Abu_Sayyaf`
	lemma around	`trig_lem_cont_0 the Abu_Sayyaf terrorist`
	lemma after	`trig_lem_cont_-1 Abu_Sayyaf terrorist kill`
	POS before	`trig_pos_+1 P-ACP DT J`
	POS around	`trig_pos_-1 J N2 VVG`
	POS after	`trig_pos_0 DT J N2`
	word before	`trig_wf_func_+1 by the Abu_Sayyaf`
	word around	`trig_wf_cont_0 the Abu_Sayyaf terrorists`
	word after	`trig_wf_cont_-1 Abu_Sayyaf terrorists killing`

We trained SVMs using these binary features with a linear kernel, producing one classifier for each target string that had more than one possible entity. Below is the break-down of target strings into ambiguous and non-ambiguous classes. For the evaluation corpus, we trained supervised classifiers for 213 of the 514 unique strings.

Corpus	*Target Strings*	*No Entities*	*One Entity*	*Ambiguous*
`Sample v2.0`	67	29	22	16
`DevData`	73	3	36	34
`Evaluation`	514	103	198	213

5 Knowledge-Based Disambiguation

Our third method for disambiguating entity mentions employed Wikipedia as a knowledge-rich resource for obtaining candidate entities — a natural fit, since the target entities were themselves drawn from it. To this end, we simply treated Wikipedia as a standard IR repository, with each article a document. The mention and surrounding context were then treated as a document query against this repository, with cosine similarities between its and the articles' TF-IDF vectors used for scoring.

The intent here is similar to the use of corpus-based WSD techniques such as the Lesk algorithm [Les86]: some form of overlap with the gloss for a given sense is used to identify the likeliest match. For example, observing a mention of `Lee` and `Jefferson Davis` in the context, the mention would be more indicative of `Robert E. Lee` than of `Bruce Lee`.

5.1 Resource Preparation

We used the October 8th, 2008 English Wikipedia dump stripped of non-content pages, such as discussions and redirects. To further reduce noise, we kept only the articles with a content area of at least 100 tokens. We then zoomed in on the primary text intended for the readers by removing elements contained in MediaWiki markup (e.g., infoboxes) and using the display values of links, instead of the canonical link target names, when available.

We then indexed the resulting text using Lucene and queried its index to identify matching articles.

We also explored generating a Lucene index over just the content in the provided Knowledge Base, as it was also derived from Wikipedia. However, its performance on the development set was poor compared to using the entire Wikipedia dump (subject to the described filtering), and we did not pursue this avenue further.

5.2 Querying and Scoring

With the resource in place, for a given mention and document pair, we employed several methods to formulate a query against it. The following variations depend on the amount of surrounding context that could be pulled in:

- the mention alone, without the surrounding context;
- the last occurrence of the mention in the text, with a span of 25 tokens to the left and to the right of the mention;
- the concatenation of all matching mentions, and their 25 token spans, in the text;
- a window of 1,000 tokens around the last mention in the text.

Preliminary results on a development set showed that using the concatenation of all occurrences of the mention and their 25 word contexts performed best, and this was used for test set queries.

After issuing the query, we filtered the returned list of Wikipedia articles and similarity scores, retaining only those covered by the dictionary. The rationale here was to match the same set of articles under consideration by the other methods.

6 Results

We evaluate our entity linking system using the scorer provided for the TAC KBP entity linking task. We report both the micro-average accuracy (average over all queries), and the macro-average accuracy (average over queries by entity).

Performance of our various components from the 2009 TAC-KBP entity linking task indicates that the context-independent dictionary works surprisingly well and that it is difficult for the context-sensitive components to improve on it (see Table 1):

	Micro			Macro		
	3904 queries	1675 KB	2229 NIL	560 entities	182 KB	378 NIL
`run1`	0.7485	0.6949	0.7887	0.6851	0.5535	0.7485
`run2`	**0.8215**	**0.8078**	0.8318	**0.7303**	**0.6737**	0.7575
`goog`	0.7789	0.6955	**0.8416**	0.7135	0.5495	0.7924
`supervised`	0.7705	0.7039	0.8205	0.6963	0.5723	0.7560
`knowledge`	0.7029	0.5272	0.8349	0.6768	0.3765	**0.8214**
`nil`	0.5710	0.0000	1.0000	0.6750	0.0000	1.0000

Table 1: Entity linking results for the 2009 TAC-KBP evaluation set (news data).

We evaluated two components for the 2010 TAC-KBP entity linking task:

1. *cascade of dictionaries* (`run1`) — an overall context-independent score for each entity, which defaults to LNRM (lower-cased normalized) in case of an EXCT (exact match) miss;
2. *heuristic disambiguation* (`run2`) — a context-dependent score, obtained by querying the longest matching entity name (based on a combination of NER, coreference, and matching Wikipedia titles) against the context-independent dictionary.

For reference, we also include the results of some of our other entity-linking components from the 2009 TAC-KBP:

1. *dictionary based on Google results* (`goog`) — the "GOOG" flavor of the dictionary;
2. *distantly supervised disambiguation* (`supervised`) — a multi-class classifier trained on Wikipedia sentences;
3. *Wikipedia knowledge* (`knowledge`) — cosine-similarity and TF-IDF and link scores from Wikipedia;
4. *NIL baseline* (`nil`) — a baseline that always returns NIL.

Both context-independent methods achieved good performance, with the cascaded dictionary (`run1`) and the Google-based dictionary (`goog`) scoring 75.85% and 77.89% on the 2009 news data. In contrast, the knowledge-based approach got only 70.27% of the answers, and while the supervised disambiguation component was able to do better than the cascaded dictionary, with 77.05%, it still fell short of the broad-coverage Google-based dictionary. The context-sensitive approach (`run2`) boosted accuracy to 82.15%.

On the 2010 training set, which consists mostly of web data, the context-dependent approach does not seem to help, with the two context-independent dictionaries giving the highest scores (see Table 2). We suspect that web data does not provide the kind of context offered by news, and that whatever contextual clues exist in its text are not captured as well by our components.

	Micro			Macro		
	1500 queries	1074 KB	426 NIL	463 entities	462 KB	1 NIL
`run1`	**0.8727**	0.8799	**0.8545**	0.8174	0.8173	**0.8545**
`run2`	0.8507	**0.8939**	0.7418	**0.8299**	**0.8301**	0.7418
`goog`	0.8520	0.8873	0.7629	0.8244	0.8245	0.7629
`supervised`	0.8393	0.8808	0.7347	0.8163	0.8165	0.7347
`knowledge`	0.7740	0.7784	0.7629	0.7235	0.7234	0.7629
`nil`	0.2840	0.0000	1.0000	0.0022	0.0000	1.0000

Table 2: Entity linking results for the 2010 TAC-KBP training set (web data).

For the 2010 evaluation set, we only provide results for our two submissions, plus the median and best results among all participants. Overall, both the cascaded dictionary and the context-sensitive approach give similar scores (see Table 3), well above the median. When it came to NILs, the cascaded dictionary consistently outperformed the context-sensitive approach (which in turn did better across non-NILs — see Table 4).

	2250 queries	750 ORG	741 GPE	751 PER
`run1`	**0.8000**	0.7507	**0.7183**	**0.9508**
`run2`	0.7933	**0.7813**	0.6849	0.9134
`highest`	0.8680	0.8520	0.7957	0.9601
`median`	0.6836	0.6767	0.5975	0.8449

Table 3: Entity linking results for the 2010 TAC-KBP evaluation set, including best and median results among all participants.

	2250 queries		750 ORG		741 GPE		751 PER	
	non-NIL	NIL	non-NIL	NIL	non-NIL	NIL	non-NIL	NIL
`run1`	0.6510	**0.9236**	0.5099	**0.9148**	0.6680	**0.8211**	0.8122	**0.9777**
`run2`	**0.7706**	0.8122	**0.6382**	0.8789	**0.8091**	0.4309	**0.8685**	0.9312

Table 4: Entity linking results for the 2010 TAC-KBP evaluation set (non-NIL vs. NIL).

This year, there was an additional entity-linking task where systems were not allowed to use the free text from the Wikipedia pages associated with the knowledge base nodes. Our `run1` system fulfills this requirement, and outperforms the highest system participating in that subtask (0.8000 vs. 0.7791), but unfortunately we did not submit to this track.

7 Conclusions and Future Work

We described the joint Stanford-UBC knowledge base population system for the entity linking task. In 2009, we developed several (mostly context-independent) approaches based on frequencies of back-links, training on contexts of anchors, overlaps of contexts with the entity's Wikipedia text, and both heuristic and supervised combinations. This year we focused on a simple context-sensitive heuristic for entity disambiguation. It performed surprisingly well on newswire data from the 2009 TAC-KBP entity linking evaluation corpus, but did less well on the 2010 TAC-KBP's web data. Since this approach appears less robust than our combined system from last year, we plan to throw it into the mix, incorporating the obvious contextual clues as features for machine learning-based techniques. For this year's new challenge, in which systems were not allowed to use the free text from Wikipedia pages associated with the knowledge base nodes, our context-independent run outperformed all participating systems (sadly, we neglected to submit it into this track).

References

[ACJ$^+$09] E. Agirre, A. X. Chang, D. S. Jurafsky, C. D. Manning, V. I. Spitkovsky, and E. Yeh. Stanford-UBC at TAC-KBP. In *TAC*, 2009.

[AE06] E. Agirre and P. Edmonds, editors. *Word Sense Disambiguation: Algorithms and Applications*, volume 33 of *Text, Speech and Language Technology*. Springer, 2006.

[FGM05] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*, 2005.

[Les86] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *SIGDOC*, 1986.

[RLR$^+$10] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning. A multi-pass sieve for coreference resolution. In *EMNLP*, 2010.