A comparison of Named-Entity Disambiguation and Word Sense Disambiguation

Angel X. Chang†, Valentin I. Spitkovsky†, Christopher D. Manning†, Eneko Agirre‡
†Computer Science Department, Stanford University, Stanford, CA, USA
‡IXA Group, University of the Basque Country, Donostia, Basque Country
{angelx,vals,manning}@cs.stanford.edu, e.agirre@ehu.eus

Abstract

Named Entity Disambiguation (NED) is the task of linking a named-entity mention to an instance in a knowledge-base, typically Wikipedia-derived resources like DBpedia. This task is closely related to word-sense disambiguation (WSD), where the mention of an open-class word is linked to a concept in a knowledge-base, typically WordNet. This paper analyzes the relation between two annotated datasets on NED and WSD, highlighting the commonalities and differences. We detail the methods to construct a NED system following the WSD word-expert approach, where we need a dictionary and one classifier is built for each target entity mention string. Constructing a dictionary for NED proved challenging, and although similarity and ambiguity are higher for NED, the results are also higher due to the larger number of training data, and the more crisp and skewed meaning differences.

Keywords: Word Sense Disambiguation, Named-Entity Disambiguation

1. Introduction

A named-entity mention may refer to multiple entities, and the process of resolving the appropriate meaning in context is called entity linking (EL) or named entity disambiguation (NED). The former terminology stresses the importance of linking a mention to an actual instance in the given knowledge-base (McNamee and Dang, 2009). We prefer the latter term (NED), which focuses on the potential ambiguity among several possible instances. It highlights the connection to the closely related problem of word-sense disambiguation (WSD), where the goal is to disambiguate open-class words with respect to a sense inventory, such as WordNet (Fellbaum, 1998).

In this paper we compare the closely related worlds of WSD and NED. In WSD, an exhaustive dictionary is provided, while in NED, one has to generate all candidate entities for a target string — a step that has been shown to be critical to success (Hachey et al., 2012), where it is called "candidate generation". In WSD very few occurrences correspond to senses missing in the dictionary, but in NED this problem is quite prevalent. We show also that ambiguity is larger for NED. On the bright side, there is a lot of potential training data for NED in the form of human-generated anchor-texts, while hand-annotated data for WSD is a scarce resource.

Section 2 introduces two popular datasets, TAC-KBP and Senseval-3, which we use to compile statistics on ambiguity and synonymy for WSD and NED in Section 3. Section 4 focuses on the performance of the Most Frequent Sense baseline, a widely used WSD baseline. This section also introduces the method to build dictionaries for NED, which is a requirement for collecting the sense frequency statistics. Section 5 explains how a supervised WSD architecture (the so-called word-expert approach) is applied to NED with good results. A longer version of this paper is available at `http://arxiv.org/abs/1603.04767`.

2. Datasets

We selected two of the most popular datasets, TAC-KBP for NED, and Senseval-3 for WSD, which, being both lexical sample[1], allow for easy comparison of a number of interesting factors.

2.1. WSD dataset

The Senseval-3 *lexical sample* dataset (Mihalcea et al., 2004) contains annotated occurrences, for 57 target words. The examples are split into train and test sets, allowing us to train supervised classifiers. Our analysis will be based on the nouns in the train part (*S3LS* for short), as named-entities are most similar to nouns. *S3LS* contains 3,593 examples for 20 nouns, an average of 180 examples for each noun. The dataset did not include multiword expressions. Note that 97.5% of the occurrences have been assigned a dictionary sense, leaving only 2.5% of the occurrences without a sense assignment.

The annotations guidelines allowed tagging of an occurrence with several mentions. This can occur when the context is not specific enough to constrain the set of possible senses to a single one. In fact, 8.3% of the examples have been labeled with more than one sense. The authors reported an Inter Tagger Agreement (ITA) of 62.8%, showing the difficulty of the task. WSD systems are evaluated on the test part, using accuracy.

2.2. NED dataset

The TAC-KBP exercise provides an inventory of target named entities, based on a subset of Wikipedia articles that had info-boxes in October of 2008. This Knowledge Base (KB) contains 818,741 entities.

Given a query that consists of a string and a document ID, the task is to determine the KB entity to which that document's string refers to (or to establish that the entity is not present in the reference KB). In other words, for each query, a system must return a KB ID (or NIL when there is no corresponding KB-instance). The string in the query can be a

[1]In WSD lexical-sample datasets refer to datasets where several selected target words are annotated in context. In contrast, all-words datasets comprise continuous text segment where all words have been annotated.

	Corpora		Dictionaries		
	WSD	NED	WSD	LNRM	HEUR
Words	20	1,162	20	1,162	1,162
NILs	0	462	0	111	94
Monosemous	0	488	0	286	331
Polysemous	20	112	20	765	737
Ambiguity	5.05	2.34	5.80	86.1	22.3

Table 1: Ambiguity rate for polysemous targets measured in the annotated corpora (left) and dictionaries (right). LNRM and HEUR refer to two NED dictionaries (cf. Section 4.1.).

single word or a multi-word. The document provides context which may be useful in disambiguating the string. All of the queries had been tagged by a team of annotators.

TAC-KBP has been running and releasing development and test data each year since 2009. We use the development and test data available in the 2010 task (*TAC10* for short), which includes examples from news and the web. The development dataset has 1,162 query strings (which we call target words for comparability with WSD, even if they are multi-token) and 5,404 examples, with an average of around 5 examples for each target word. We will see (Section 3) that in many cases the queries refer to the same entity, even if using a different string mention. Note that, contrary to WSD the development data cannot be used to train a supervised WSD system, only to tune hyper-parameters. Supervised NED systems are trained using external data Section 5.

Since the inventory of entities used in TAC-KBP is incomplete, it is common for target words to be annotated as NIL (49% of total examples), in contrast to WSD (less than 3%). In all examples there is a single ID per target word, that is, there are no cases where multiple entities are referred to, in contrast to WSD (8% of cases). ITA was reported separately for each entity type. It was higher than for WSD in all cases: organizations (92.98%), people (91.53%), and locations (87.5%). NED systems are evaluated using accuracy. In our case, we will focus our results on the examples which do have a referent in the KB (so-called non-NIL cases).

3. Ambiguity and Synonymy

Ambiguity refers to the number of referents (instances for NED, concepts for WSD) that a target word has. In the case of WSD, this is also known as the polysemy rate. Synonymy rate, or synonymy for short, refers to the number of lexicalizations (strings) that each instance or concept (NED or WSD, respectively) can have. Estimating ambiguity and synonymy rate is not trivial, as it assumes that a complete inventory of lexicalizations and referents (concepts and instances) exists, as well as a complete relation between lexicalizations and referents. Unfortunately no such inventories exist. In the case of WSD, the dictionary provided by WordNet is fairly complete, as attested in *S3LS*, where the annotators missed a sense in only 2.5% of the examples. In the case of NED both constructing an inventory of all relevant instances and collecting the possible realizations for those instances is an open-ended problem (Hachey et al., 2012). Even for the limited subset of instances listed in Wikipedia, there is no standard dictionary of entities which lists all possible lexicalizations and their respective instances.

Given those limitations, we present two main estimations of ambiguity and synonymy rate: one based on tagged corpora, where we record the ambiguity and synonymy rate attested in the annotated corpora only, and another one based on dictionaries. For the latter, we will use WordNet for WSD, and several versions of Wikipedia-derived dictionaries (cf. Section 4.1) for NED.

3.1. Ambiguity and synonymy in corpora

For each target word in a dataset, we estimate ambiguity as the number of different instances or concepts (NED and WSD, respectively) associated to it by human annotators. The left columns of Table 1 report the number of target words, how many of them have no example with a KB instance or concept attested in the corpus (NIL row), how many are monosemous (single KB instance or concept), how many are polysemous, and the mean ambiguity for polysemous targets.

In NED, many target strings tend to be highly skewed, in the sense that they usually refer to one specific entity. This fact is reflected by the large number of strings which refer to a single entity in the annotated corpus. We will show that actual polysemy is much higher (e.g., according to our dictionaries — see Section 3.2), since many of the possible entities do not appear in annotated dataset.

Another quantity that sheds light on the complexity of a disambiguation task, in a manner that is complementary to ambiguity, is the number of different lexicalizations (words or multi-words) that can be used to name a particular entity or concept: its *synonymy* rate, in WSD terms. The left columns in Table 2 tabulate the synonymy rate attested in the annotated datasets, together with the breakdown, for each concept or entity in the dataset, of how many of those are not lexicalized, have a single lexicalization or multiple lexicalizations. The synonymy rate is computed for those referents having multiple lexicalizations.

The 101 referents present in the WSD dataset get an average of 3.34 lexicalizations. In the case of NED, most entities are associated with only a single string (85%), and the synonymy rate is lower, 2.49. This could be an artifact of how the organizers constructed the dataset, since their procedure was to first choose ambiguous strings and then find documents containing different meanings (entities) for these mentions — as opposed to first choosing entities and then querying string variants of names. The annotators thus did not search for alternative lexicalizations (synonyms) to be used as query strings. As was the case with ambiguity, the actual synonymy rate in dictionaries for NED is much higher (see Section 3.2).

3.2. Ambiguity and Synonymy in Dictionaries

The right columns in Table 1 reports ambiguity figures for two of the dictionaries we constructed for NED, **LNRM** and **HEUR**, to be introduced in Section 4.1, as well as the WSD dictionary, WordNet. **HEUR** covers more strings than **LNRM** and has fewer monosemous entries yet lower polysemy. **HEUR** retains more references from Wikipedia and, simultaneously, substantially reduces ambiguity.

	Corpora		Dictionaries		
	WSD	NED	WSD	EXCT	HEUR
Referent	101	1,239	114	1,239	1,239
No Lex.	0	0	0	279	296
Single Lex.	30	1,053	34	0	0
Multiple Lex.	71	186	80	960	943
Synonymy	3.34	2.49	3.35	210.38	46.37

Table 2: Synonymy rate of referents (instances or concepts), measured in the annotated corpora (left) and dictionaries (right). EXCT and HEUR refer to two NED dictionaries (cf. Section 4.1).

In contrast to WSD, where the ambiguity figures attested in the corpora and dictionaries are similar, the ambiguity figures for NED in dictionaries are much higher than those attested in the annotated corpus (compare to Table 1). This comes as no surprise, since the gold standard severely underestimates true ambiguity by focusing exclusively on entities that are mentioned in a target dataset. In contrast to WSD, NED's true ambiguity usually remains unknown, as its determination would require a manual inspection of all strings and entities, and an exhaustive search for examples of usage in actual text. Although the NED dictionaries have good quality and come close to covering all entities in the gold standard for known target words (cf. Section 4.2) a manual inspection showed that they also contain incorrect entries. Therefore, we suspect that actual ambiguity maybe be slightly lower than our estimate obtained with the **HEUR** dictionary.

Table 2 shows synonymy figures for both the **EXCT** and **HEUR** dictionaries, as well as the WSD dictionary, WordNet[2]. In the case of NED, entities corresponding to NILs were not covered by the dictionary (tallied under the *No Lex.* heading), and all of the remaining entities were lexicalized by a large number of strings (none by just one), in contrast to the situation in WSD. The **EXCT** dictionary had, on average, 210 strings per entity, which is reduced to 46 with heuristics.

Although the high estimates reflect the comprehensive coverage afforded by our dictionaries, they do not reveal true levels of synonymy, which would require hand-checking all entries, as before. Nevertheless, these figures illustrate, at a high level, another important difference between NED and WSD: the synonymy rate is much lower for WSD, and finding all lexicalizations for an instance in NED is an open-ended problem.

4. Most Frequent Sense

The most frequent sense (MFS) is a high performing baseline widely used in WSD. A MFS system returns the sense with the highest frequency, as attested in training data. It presupposes that, for a given target word, all possible referents are known, and then it estimates their frequency from annotated data. This is straightforward for WSD, as every target word has a canonical entry in the dictionary (e.g. WordNet). In the case of our dataset, MFS statistics for WSD are directly gathered from the training data made available.

In the case of NED the work is complicated by the lack of a dictionary listing canonical entries. In fact, there are several ways to construct such a dictionary, as we will see next. In any case, we take the opportunity to also gather sense frequency data from anchors in Wikipedia and the web.

4.1. Constructing dictionaries for NED

Generating the set of entities that a specific string could refer to is a critical module in many NED systems. It can range from using Wikipedia titles as lexicalizations, to using all anchor texts, and several other related techniques (Hachey et al., 2012). All of these techniques can introduce incorrect lexicalizations, e.g. "dinner" being used in an anchor to refer to a specific dish like Ceviche. In addition, the dictionary also gathers the counts of anchors, which are used to estimate the most frequent sense.

The dictionary is represented as a set of weighted pairs — an exhaustive enumeration of all possible string-entity combinations, with corresponding strengths of association. We constructed this resource from all English Wikipedia pages (as of the March 6th, 2009 data dump) and many references to Wikipedia from the greater web (based on a subset of a 2011 Google crawl). Individual string-entity co-occurrences were mined from several sources:

- Article titles that name entities, e.g., "Robert Redford" from `http://en.wikipedia.org/wiki/Robert_Redford`.[6] Many title strings had to be processed, e.g., separating trailing parentheticals, like "(athlete)" in `Jonathan_Edwards_(Athlete)`, and underscores, from names proper.
- Titles of pages that redirect to other Wikipedia pages, e.g., "Stanford" for the article `Stanford_University`, redirected from the page `Stanford`.
- Titles of disambiguation pages that fan out to many similarly-named articles, e.g., linking "Stanford" to `Stanford_University`, `Aaron_Stanford` or `Stanford,_Bedfordshire`, among other possibilities, via `Stanford_(disambiguation)`.
- Anchor text, e.g, we interpret the fact that Wikipedia editors linked the two strings *"Charles Robert Redford"* and *"Robert Redford Jr."* to the article `Robert_Redford` as a strong indication that both

[2]The WSD column for dictionaries includes all concepts of target strings, while the NED column for dictionaries only includes entities in the corpora.

[3]I.e., either (a) the string is an acronym for the title; or (b) the title is an acronym for the string.

[4]I.e., either (a) the strings are the same; or (b) both strings have length less than or equal to six, with an edit distance exactly equal to one; or (c) the ratio between edit distance and string length is less than or equal to 0.1.

[5]I.e., if (a) the number of total links to the page (both inside Wikipedia and from the external web) is no more than ten; or (b) the number of times the string links to the page is no more than one; or (c) the score is no more than 0.001.

[6]In the remainder of this article, we will use the following conventions: "string" for a string that can be used to name an entity (e.g., "Robert Redford") and suffixes of English Wikipedia URLs, without the prefix `http://en.wikipedia.org/wiki/`, as short-hand for corresponding articles (e.g., `Robert_Redford`).

0.9976	`Hank_Williams`	`w:756/758`	`W:936/938`
0.0012	`Your_Cheatin'_Heart`		`W: 2/938`
0.0006	`Hank_Williams_(Clickradio_CEO)`	`w: 1/758`	
0.0006	`Hank_Williams_(basketball)`	`w: 1/758`	
0	`Hank_Williams,_Jr.`		
0	`Hank_Williams_(disambiguation)`		
0	`Hank_Williams_First_Nation`		
0	`Hank_Williams_III`		

Figure 1: Sample from the **EXCT** dictionary, listing all articles and scores for the string *"Hank Williams"*. Final column(s) report counts from Wikipedia (`w`:x/y) and the web crawl (`W`:u/v), where available.

0.9524	`Hank_Williams`		`W:20/21`
0.0476	`I'm_So_Lonesome_I_Could_Cry`		`W: 1/21`
0	`Hank_Williams_(Clickradio_CEO)`		
0	`Hank_Williams_(basketball)`		
0	`Hank_Williams_(disambiguation)`		

Figure 2: Sample from the **LNRM** dictionary (note that contributions already in **EXCT** are excluded), for strings with forms like $l($*Hank Williams*$) =$ *hankwilliams* $= l($*HANK WILLIAMS*$) = l($*hank williams*$)$, etc.

0.6316	`Tank_Williams`	`w:12/12`	
0.3158	`Hank_Williams`		`W:6/7`
0.0526	`Your_Cheatin'_Heart`		`W:1/7`

Figure 3: Sample from the **FUZZ** dictionary for strings one byte way from *"Hank Williams"*, including *"Tank Williams"*, *"Hanks Williams"*, *"hankwilliam"*, etc. (note that contributions already in **EXCT** or **LNRM** are excluded).

could refer to "Robert Redford." We use the number of links connecting a particular string with a specific entity as a measure of the strength of association.

Note that our dictionary spans not just named entities but also many general topics for which there are Wikipedia articles. Further, it transcends Wikipedia by including anchors (i) from the greater web; and (ii) to Wikipedia pages that may not (yet) exist. For the purposes of NED, it could make sense to discard all but the articles that correspond to named entities. We keep everything, however, since not all articles have a known entity type, and because we would like to construct a resource that is generally useful for disambiguating concepts. Our dictionary can disambiguate mentions directly, simply by returning the highest-scoring entry for a given string. The construction of this dictionary is explained with more details in (Spitkovsky and Chang, 2012).

We developed several variants of the dictionary: **EXCT**, **LNRM**, **FUZZ** and **HEUR**. For a given string-article pair, where the string has been observed as the anchor-text of a total of y inter-Wikipedia and v external links, of which x (and, respectively, u) pointed to a page that is represented by the article in the pair, we set the pair's score to be a ratio $(x + u)/(y + v)$. We call this dictionary exact (**EXCT**), as it matches precisely the raw strings found using the methods outlined above. For example, Figure 1 shows all eight articles that have been referred to by the string *"Hank Williams."* Note that this dictionary does no filtering: removal of undesired target Wikipedia pages (such as disambiguations) will be done at a later stage (see below).

Regarding **LNRM**, we form the lower-cased normalized variant $l(s)$ of a string s by canonicalizing Unicode characters, eliminating diacritics, lower-casing and discarding any resulting ASCII-range characters that are not alphanumeric. If what remains is the empty string, then s maps to no keys; otherwise, s matches all keys k such that $l(s) = l(k)$, with the exception of $k = s$, to exclude the original key (which is already covered by **EXCT**). Figure 2 shows a subset of the **LNRM** dictionary, with combined contributions from strings k that are similar to (but different from) s.

We define the **FUZZ** dictionary via a metric, $d(s, s')$: the byte-level UTF-8 Levenshtein edit-distance between strings s and s'. If $l(s)$ is empty, then s maps to no keys once again; otherwise, s matches all keys k, whose $l(k)$ is also not empty, that minimize $d(l(s), l(k)) > 0$. This approach excludes not only $k = s$ but also $l(k) = l(s)$, already covered by **LNRM**. For example, for the string *Hank Williams*, there exist keys whose signature is exactly one byte away, including *Tank Williams*, *Hanks Williams*, *hankwilliam*, and so forth. Figure 3 shows the three articles — two of them already discovered by both **EXCT** and **LNRM** dictionaries — found via this fuzzy match.

For disambiguating a target word with the MFS, we just use the dictionary to select a highest-scoring entity, using the ratios introduced above. We prepared several extensions to the **EXCT** dictionary, using two strategies. The first one is the **dictionary cascade** as follows:

- The **LNRM** cascade first checks the **EXCT** dictionary, returning its associated entities and scores if the string has an entry there, and defaulting to **LNRM** dictionary's results if not;
- The **FUZZ** cascade also first checks the **EXCT** dictionary, backs off to the **LNRM** dictionary in case of a miss, but then finally defaults to the contents from the **FUZZ** dictionary.

	Rule	*Example*
1	Discard disambiguation pages.	Discard: $\star \rightarrow$ `Hank_Williams_(disambiguation)`
2	Discard date pages.	Discard: $\star \rightarrow$ `2000`
3	Discard list-of pages.	Discard: $\star \rightarrow$ `List_of_cheeses`
4	Discard pages only suggested by **FUZZ**, unless:	Discard: *MND* $\rightarrow$ `MNW`
	· string and title could be an acronym pair;[3]	Keep: *NDMC* $\rightarrow$ `National_Defense_Medical_Center`
	· string is a substring of the title;	Keep: *DeLorean Motor* $\rightarrow$ `DeLorean_Motor_Company`
	· string is very similar to the title.[4]	Keep: *Chunghua Telecom* $\rightarrow$ `Chunghwa_Telecom`
5	Discard articles supported by few links,[5] unless:	Discard: *Washington* $\rightarrow$ `Tacoma,_Washington`
	· article may disambiguate the string;	Keep: *CNS* $\rightarrow$ `Szekler_National_Council`
	· string is the title of the page.	Keep: *Chunghwa Telecom* $\rightarrow$ `Chunghwa_Telecom`

Table 3: Rules of the heuristic combination dictionary (**HEUR**).

For the sake of simplicity, we will use **LNRM** and **FUZZ** to refer to the respective cascades in the rest of the paper.

Our second strategy is a **heuristic combination** (**HEUR**): it combines suggestions from the original (**EXCT**, **LNRM**, and **FUZZ**) dictionaries while also filtering out some of the noise. Its goal is to retain good suggestions without drowning in obviously bad choices: since many titles suggested by the **FUZZ** dictionary are noisy, we only include those for which there is additional evidence of relevance (for instance, if the suggestion is an acronym for the string). Similarly, if a string has been used to link to an article only a few times, the connection between them may not be reliable, calling for more evidence before the entity could be accepted as a possible referent for the string. Naturally, we also discard articles that are clearly not real entities — such as disambiguation pages, "list of" pages, and pages of dates — using additional features collected with the core dictionary. Table 3 summarizes our complete list of heuristics, which was finalized by consulting a development dataset.

Heuristic combinations can yield dictionaries that are considerably smaller than full cascades. For example, for the string *ABC*, the **EXCT** dictionary offers 191 options, the **LNRM** cascade 253, and the **FUZZ** cascade 3,527. But with the above-mentioned filters, the number of candidate Wikipedia titles can be reduced to just 110. We already mentioned in Section 3 Tables 1 and 2 several key statistics for some of the dictionaries.

4.2. Developing MFS for NED

Table 4 shows development results of MFS for several NED dictionaries (*News* subset of development data). The best results are for the heuristic dictionary. We measured the performance of the HEUR dictionary when using only Wikipedia counts and only the web crawl, with performance drops in both cases, 0.651 and 0.676. The table also includes GOOG, the result of querying Google with the target word and returning the Wikipedia article with highest rank, which performs slightly worse than our dictionaries.

If the dictionary fails to recognize that a given string could refer to a particular entity, then, the disambiguation system (MFS or otherwise) will not be able to return that entity. Thus, the dictionary introduces a performance bottleneck in WSD and NED.

	Accuracy	Oracle
EXCT	0.694	
LNRM	0.695	91.6
FUZZ	0.713	94.1
HEUR	0.721	91.9
GOOG	0.696	

Table 4: Development results for MFS dictionaries for NED, including oracle results.

The rightmost column of Table 4 shows the skyline results that could be attained, for each dictionary, by an oracle, choosing the best possible entity available to each system. This skyline is computed, as for the MFS, for non-NIL cases, that is, for those cases where the gold standard evaluation dataset has been annotated with an entity in the KB. For **LNRM** and **HEUR** cascades, gold standard entities are among the suggestions of the dictionary around 92% of the time. It is not 100% because the dictionary misses the association between some target words and entities in the KB, that is, even if the target word is in the dictionary and the target entity is in the KB, the dictionary entry for the target word misses to make reference to the target entity. Note that the skyline for WSD in the dataset we mentioned is 100%, as the dictionary contains all necessary associations between senses and target words.

A higher bound does not mean necessary a better performance. For example, **FUZZ** yields the overall largest number of possibilities, but has lower realized performance than **HEUR**. Regarding the performance of the supervised classifier, experiments in development data showed that the best performance was attained using the HEUR dictionary, perhaps due to the substantially higher ambiguity in the FUZZ dictionary.

5. Supervised Disambiguation

We followed a mainstream WSD approach (word-expert), training supervised classifiers for all target strings. In the case of WSD, this is straightforward, as we use the training data provided with the S3LS dataset. For NED, we proceed as follows: For every string in the **HEUR** dictionary, we first identify the entities to which it may refer. We then

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

anchor text		Abu_Sayyaf
lemmas in the span		terrorist
		kill
		...
lemma for N/V/A		be
in a 4 token window		bomb
around the anchor text		kill, people, terrorist
lemma and word for N/V/A	noun (lemma)	SuperFerry
before the anchor text	noun (word)	SuperFerry
	verb (lemma)	bomb
	verb (word)	bombed
lemma and word for N/V/A	adjective (lemma)	bad
after the anchor text	adjective (word)	worst
	noun (lemma)	terrorist
	noun (word)	terrorists
	verb (lemma)	kill
	verb (word)	killing
bigrams around anchor text	lemma before	the Abu_Sayyaf
	lemma after	Abu_Sayyaf terrorist
	POS before	DT J
	POS after	J N2
	word before	the Abu_Sayyaf
	word after	Abu_Sayyaf terrorist
trigrams around anchor text	lemma before	by the Abu_Sayyaf
	lemma around	the Abu_Sayyaf terrorist
	lemma after	Abu_Sayyaf terrorist kill
	POS before	P-ACP DT J
	POS around	J N2 VVG
	POS after	DT J N2
	word before	by the Abu_Sayyaf
	word around	the Abu_Sayyaf terrorists
	word after	Abu_Sayyaf terrorists killing

Figure 4: Example training context and features extracted from Wikipedia's article for `SuperFerry`.

gather the context of all anchors where the string occurs in Wikipedia articles. To ensure that our training data is natural language (and not, e.g., lists or tables), we only include text marked as paragraphs (i.e., enclosed between HTML tags `<P>` and `</P>`). The relevant training subset for a target string then consists of example contexts with anchor texts containing the string.[7] We take spans of up to 100 tokens to the left and another 100 to the right.

Given this training data, we applied standard machine learning techniques to perform supervised disambiguation of entities. We trained a maximum entropy multi-class classifier[8] for each target string (Manning and Klein, 2003). Then, given a mention of the target string in the test data, we applied its classifier to the context of the mention, and returned the corresponding article. We did not construct classifiers for strings whose training data maps to a unique entity. Instead, in those cases, a default classifier falls back to **LNRM** cascade's output.

From each context, we extracted features (see Figure 4) commonly used for supervised classification in the WSD setting (Agirre and Lopez de Lacalle, 2007; Zhong and Ng, 2010):

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

5.1. Variations

Over the course of developing our system, we tested several variations of the core algorithm:

Classifier: We tried maximum entropy models (MAXENT) and support vector machines (SVM).

Dictionary: A dictionary influences supervised classification in two places. First, when building the training data, to filter example spans selected for training. And second, as a backup ranker, for cases when a classifier is not trained, due to a lack of examples. In both the filtering stage and the back-off stage, we compared using the **HEUR** dictionary in place of the **LNRM** cascade.

Span: In addition to training with contexts of (up to) 100

[7]The target string is a substring of the anchor text after case normalization.

[8]ℓ_2-regularization

default	0.7707
SVM	0.7463
LNRM (filtering)	0.7528
HEUR (back-off)	0.7827
SENT	0.7582
PARA	0.7582
SENSE	0.8090

Table 5: Performance of the supervised classifier on the *news* subset of development data, in several variants (cf. Section 5.1).

tokens to the left and right of a string, we also tried single-sentence and full-paragraph spans (the **100**, **SENT** and **PARA** variants).

Match: When gathering examples for a target string, we made sure that the anchor text contains this string (the **LEX** default). Alternatively, we could allow additional examples, ignoring anchor text mismatch (the **SENSE** variant): given the entities that a dictionary lists for the target string, we include as training examples all contexts that apply to these entities, regardless of their anchor text. In this variant, the target string is simply treated as another feature by the classifier. If a test example's string does not match any of the anchor text seen in training, then features that include the target string (i.e., its unigram, bigram, and trigram features) will not fire. Classification will then depend on features describing the rest of the context: a classifier could still give a high score, but only if surrounding words of a span carry a strong enough signal for an entity. This approach may allow us to classify aliases for which there isn't exact training data, provided that our filtering dictionary yields a precise list of potential entities corresponding to a target string.

5.2. Developing the supervised system for NED

Table 5 shows performance on the *news* subset of development data for several variants of our supervised classifier. The first row corresponds to default parameters; the rest represent a greedy exploration of the space of alternatives. Each additional row specifies a setting that differs from the first row in one parameter (cf. Section 5.1).

Classifier: The second row shows the results of `SVMlight` binary classifiers (Joachims, 1999), which performs worse than our default classification algorithm (maximum entropy with ℓ_2-regularization), MAXENT (Manning and Klein, 2003). We did not tune any of the available parameters, since we were interested in out-of-the-box performance of all methods (for SVMs, we used a linear kernel with a cost of 0.01).

Dictionary: The third row shows that the **LNRM** cascade generates worse training examples than our default dictionary combination, **HEUR**. We do not show results for other dictionaries, which yield too many candidates (without improving precision). The fourth row shows that **HEUR** also performs better than **LNRM** when used as a back-off dictionary, improving over the default.

Span: Rows five and six show that supervised classification performs equally well using either sentences or paragraphs (but that the best results are obtained using left and right 100 tokens). One reason for similar performance is that most paragraphs are not marked correctly. In addition, in Wikipedia, a paragraph often comprises of a single sentence. A fixed span of tokens to each side of a mention may extend beyond sentence boundaries, providing more context to help with disambiguation.

Match: The last row shows that using all examples that refer to an entity (**SENSE**) improves over the default approach, which uses only a subset of examples that contain the target string (**LEX**).

	WSD	NED
MFS	55.2	74.9
SUPERV.	72.9	84.5

Table 6: Test results for the MFS baseline and supervised classifiers on WSD and NED.

6. Test Results

Table 6 reports the results on test data for MFS and the supervised classifiers. In the case of WSD, we report the results of the best participant in the Senseval task. For NED, the MFS results corresponds to the HEUR dictionary.

The results for NED are higher than for WSD. Although absolute error reduction is larger for WSD, the relative error reduction is very similar, 40% for NED, and 38% for WSD. Note that the best results in the TAC-KBP competition in 2009 was 80.6 (Lehmann et al., 2010), lower than our word-expert approach.

7. Discussion

We have shown that it is possible to construct an effective dictionary for NED, covering between 92% and 98% of the manually annotated string-entity pairs (for development data). Although the ambiguity in such dictionaries varies, it tends to be higher than for the gold standard in NED and also for a typical WSD dataset. Since a similar phenomenon is also observed with synonymy, one might expect NED to pose a more difficult problem than WSD; nevertheless, we observed the opposite effect in practice. Our popularity-based dictionary heuristic performs even more strongly than the MFS baseline in WSD (75% in our blind NED evaluation, compared to 55%); supervised system variants also score much higher (84% *vs.* 73%). Of course, comparing evaluation numbers across tasks requires extreme caution. Nonetheless, we suspect that qualitatively large difference in performance here indicates that NED has larger numbers of training data, compared to WSD, and also the fact that clarity and definiteness of disambiguating an entity reference to a particular person/organization contrasts with the common difficulty of doing this when there are overlapping/unclear senses for WSD. The later also explains the lower inter-annotator agreement rate for WSD compared to NED.

High ambiguity and synonymy, together with the large volumes of text data, make NED computationally more demanding than WSD, both in the scope of regular memory and disk storage capacities, as well as speed and efficient processing requirements. Our approach in particular could invoke training of potentially millions of classifiers, requiring significant engineering effort. But since each classifier can be trained independently, parallelization is easy.

The number of cases where annotators could not assign an entity (NILs) is significantly higher in NED than in WSD (around 50% compared to just 1.7%). Ambiguity and synonymy, according to gold standards, are substantially lower than in WSD (with average ambiguities of around 2 *vs.* 5, and average polysemies of 2 *vs.* 4). These statistics are misleading, however, since actual ambiguity and synonymy in NED are more extreme, according to our dictionaries (discussed in Section 3.2). Finally, inter-annotator agreement for NED is higher — between 87% and 93% — compared to 72% in WSD.

We can summarize the main differences between WSD and NED as follows: (1) ambiguity, synonymy and incidence of dictionary misses (NILs) are all higher for NED than for WSD; (2) the NED task appears better-defined, as signaled by higher inter-annotator agreement than in WSD; (3) the skew of frequencies is more extreme for NED, with MFS consequently presenting an even stronger baseline there than in WSD; (4) the high number of training instances available to NED enables better supervised training, allowing comparable systems (same preprocessing, feature sets, and classifier) to perform better in NED than in WSD; (5) the relative error reduction between the performance of a supervised classifier and the MFS baseline is strikingly similar in both WSD and NED.

In addition we show the following: (6) a dictionary for NED can provide good coverage and MFS results, where web crawl counts and Wikipedia counts add up for best results; (7) the high ambiguity of mentions encountered by NED makes a typical word expert approach more computationally expensive, but still feasible.

8. Conclusions and future work

We highlighted many connections between WSD and NED. In WSD settings, dictionaries are provided, but NED involves constructing possible mappings from strings to entities — a step that (Hachey et al., 2012) showed to be key to success. The resulting dictionaries exhibit very high synonymy and ambiguity (polysemy) yet still do not cover many occurrences that ought to be tagged by a NED system, making the task appear more difficult, in theory, compared to WSD. But in practice, the opposite seems to be the case, due to actual mentions being more heavily skewed towards popular entities than in WSD, a plethora of available training data in the form of human-entered anchor-texts of hyperlinks on the web, and higher inter-annotator agreement, which indicates more crisp differences between possible shades of meanings than in WSD. As a result, both popularity-based dictionary lookups (MFS heuristics) and supervised classifiers, which are traditional WSD architectures, perform better for NED than for WSD.

Our study has general considerations, but some statistics are related to the two datasets under study. In the future, we would like to extend our study using datasets which include all mentions in full documents (Snyder and Palmer, 2004; Hoffart et al., 2011). A longer version of this paper is available at `http://arxiv.org/abs/1603.04767`.

9. Acknowledgements

We thank Oier Lopez de Lacalle and David Martinez, for the script to extract features, as well as Daniel Jurafsky and Eric Yeh, for their contributions to an earlier participation in TAC-KBP.

Angel X. Chang has been supported by a SAP Stanford Graduate Fellowship; Valentin I. Spitkovsky has been partially supported by NSF grants IIS-0811974 and IIS-1216875 and by the Fannie & John Hertz Foundation Fellowship. We gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

10. Bibliographical References

Agirre, E. and Lopez de Lacalle, O. (2007). UBC-ALM: Combining k-NN with SVD for WSD. In *SemEval*.

Christiane Fellbaum, editor. (1998). *WordNet: An Electronic Database*. MIT Press.

Hachey, B., Radford, W., Nothman, J., Honnibal, M., and Curran, J. (2012). Evaluating Entity Linking with Wikipedia. *Artificial Intelligence*, 194:130–150, January.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *EMNLP*.

Joachims, T. (1999). Making large-scale SVM learning practical. In Bernhard Schölkopf, et al., editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Lehmann, J., Monahan, S., Nezda, L., Jung, A., and Shi, Y. (2010). LCC approaches to knowledge base population at TAC 2010. In *TAC*.

Manning, C. D. and Klein, D. (2003). Optimization, maxent models, and conditional estimation without magic. In *HLT-NAACL*.

McNamee, P. and Dang, H. (2009). Overview of the TAC 2009 Knowledge Base Population track. In *TAC*.

Mihalcea, R., Chklovski, T., and Kilgarriff, A. (2004). The Senseval-3 English lexical sample task. In *Senseval*.

Snyder, B. and Palmer, M. (2004). The English all-words task. In *Senseval*.

Spitkovsky, V. I. and Chang, A. X. (2012). A cross-lingual dictionary for English Wikipedia concepts. In *LREC*.

Zhong, Z. and Ng, H. T. (2010). It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL: System Demonstrations*.