UMNDuluth at SemEval-2016 Task 14: WordNet's Missing Lemmas

Jon Rusert & Ted Pedersen
Department of Computer Science
University of Minnesota
Duluth, MN USA
{ruse0008,tpederse}@d.umn.edu

Abstract

This paper presents a solution to Semeval 2016 Task 14 which asks for a system that is able to insert new lemmas into WordNet. Our system aims to do this by overlapping words in the definitions of the to-be-inserted lemma and all senses in WordNet. This paper includes the results of our system and also includes the baseline provided by Task 14, with our system scoring higher than the random baseline, and lower than the first word baseline.

1 Introduction

Semeval 2016 Task 14 called for a system that could help enrich the WordNet taxonomy with new words and their senses. This translates to inserting new lemmas and senses that were previously not in WordNet into their (human perceived) correct place. The system was also to determine whether a sense would merge into the chosen synset or attach itself as a new hyponym. Task 14 allowed for one of two types of systems:

1. A resource-aware system, which could use any dictionary, or
2. Constrained, which used any resource other than a dictionary.

We opted with the former, resource-aware. While any dictionary could be used, we chose to use the definitions provided in the data set from Wiktionary, along with definitions from WordNet.

2 Methods

Our system solves the given problem in four steps:

1. Pre-processing: Acquire all necessary data from WordNet and store it in one step. The data needed includes each word's definitions, hypernyms, hyponyms, and synsets.
2. Overlaps: Score each sense on how well it matches each new lemma.
3. Refining chosen sense: Verify that the sense chosen in overlaps was more deserving than the other senses of the same lemma.
4. Determining attach or merge: Decide whether or not the new lemma should be attached to the synset of the chosen sense, or merged into it.

2.1 Pre-processing

As the implementation of our system was underway, it was clear that the system would be making a large amount of calls to WordNet. As we accessed more and more data from WordNet[1], our program took longer to finish each time which created a problem for testing out changes quickly. In response, the pre-processing method was created.

Pre-processing aims to consolidate all calls to WordNet in the beginning of the program, so no duplicate calls need to be made. It does this by first obtaining all nouns and verbs from WordNet and storing them in their respective arrays (one array for nouns and one for verbs). Pre-processing

[1] http://search.cpan.org/dist/WordNet-QueryData/QueryData.pm

then iterates through each word and retrieves each sense of each word, since the senses are what will determine which synset the new lemma will be merged or attached to later on. The senses are stored in a separate array, which is iterated through, one by one, in the Overlap step to obtain a score for each sense. Next, it iterates through each sense and obtains that senses gloss. Pre-processing cleans each gloss by making all letters into lowercase, removing punctuation, and also removing this list of common stop words *(the—is—at—which—on—a—an—and—or—up)* from each gloss. This list of stop words was determined by finding common, less helpful words in the trial/test data. These stop words were found by outputting what words were being overlapped, and these appeared the most frequently even though they rarely added positively to the overlaps scores. It then stores the cleaned gloss in a hash that maps the gloss to the corresponding sense. Finally, Pre-processing obtains the hypernyms, hyponyms, and synsets for each sense and stores them in their respective hashes (hypernyms, hyponyms, and synsets).

2.2 Overlap

The Overlap step is the main step in our system for determining where the new lemmas would be inserted into WordNet. Ideas were borrowed from both (Lesk, 1986) and Extended Gloss Overlaps (Banerjee and Pedersen, 2003).

2.2.1 Lesk

Lesk overlaps work by comparing two words' definitions and seeing if words in those definitions overlap onto one another. Words that share more overlaps score higher with the Lesk algorithm and therefore are more similar. However, one weakness with the Lesk algorithm is that different dictionaries might define even the same word differently, which means the number of overlaps is highly dependent on the dictionary used.

2.2.2 Extended Gloss Overlaps

To address the room for error in the Lesk overlaps, Extended Gloss Overlaps (EGO) incorporate not only the definitions of each word being compared, but also the definitions of the hypernyms and hyponyms of each word. EGOs use WordNet to retrieve the hypernyms/hyponyms and their respective definitions for scoring. It was after EGOs that our system of scoring and calculating overlaps is based on.

2.2.3 Overlap Step

Our Overlap step works by iterating through each sense obtained from WordNet and creating an expanded sense by adding information from each sense. The expanded sense is then compared to the to-be-inserted lemmas creating a score to determine how alike the terms are. It should be noted that only corresponding parts of speech were compared as to improve time and not cause nouns to be mapped to verbs and vice versa.

For each sense to be compared, the expanded sense was created. First the sense's gloss was obtained from the hash initialized in pre-processing. Next the sense's immediate hypernyms and their glosses were retrieved and added to the expanded sense. Likewise, the sense's immediate hyponyms and their glosses were retrieved and added to the expanded sense. Finally, the sense's corresponding synset and their corresponding glosses were retrieved and added to the expanded sense. Next before any word overlaps could be processed, the new lemma's gloss needed to be cleaned up.

To provide clarity, we will act as if *ink* (taken from provided trial data) is being inserted into WordNet. For reference *ink*'s provided definition was "Tattoo work." The lemma was cleaned up following the same steps as the WordNet glosses followed in the pre-processing step. *Ink*'s definition would now become "tattoo work", since all letters are made lowercase. However, since *ink* did not contain any stop words on the list, none were removed.

Now the system steps through each word in the lemma's gloss and checks for overlaps in the glosses of the expanded sense's gloss, each hypernym's gloss, each hyponym's gloss, and finally each synset's gloss. If the word being checked is part of the lemma of each sense, it receives a bonus score. The bonus score was originally set to (10 * the length of the lemma) but was later changed to (2 * the length of the lemma). This bonus was limited to compound words of at most two words. The decision to limit the length of compound words was arrived at since larger compounds like *Standing-*

on-top-of-the-world would score higher than *World-wide* just because they were much longer compounds, even though they occur less often.

Since *ink*'s definition contains the word *tattoo*, any sense with *tattoo* in its lemma will receive the bonus. This means that *tattoo#n#:.* (i.e. any noun sense of *tattoo*) would receive a bonus. The same holds true for *work#n#:.* (i.e. any noun sense of *work*). The overlapping of words were also weighted by the number of characters present in those words (or more simply length of those words), so longer words carried a heavier weight in the score than shorter ones. As with *ink*, when the word *tattoo*, in the definition of ink, overlaps with another compared word it adds 6 to the score since *tattoo* contains 6 letters, whereas *work* would only add 4 to the score.

The final score of the sense was calculated by dividing the number of overlaps by the total length of words from the new term.

$$\begin{aligned} score = (&SenseLaps + HypeLaps \\ &+ HypoLaps + SynsLaps \\ &+ BonusLapsTotal)/GlossLength \end{aligned} \quad (1)$$

The sense with the highest score at the end was presumed to be the chosen sense to either attach or merge to.

Our system determined that *ink* belonged to *tattoo#n#3* whose definition from WordNet was, "the practice of making a design on the skin by pricking and staining". Since *ink* had a short definition provided from Wiktionary, the largest score came from the fact that *tattoo* gained the bonus score from overlapping with the definition. The correct answer provided in the key was *tattoo#n#2*, the reason for the differences was most likely the fact that our system did not identify present participle words, since *tattoo#n#2* contained the word "tattooing".

2.3 Refining the chosen sense

Now that the sense had been chosen, a new measure was implemented to make sure that, in fact, the correct sense had been chosen. This step was added since it was often the case that the first sense of a word was a better choice, however, a different sense of the same word would tie causing it to replace the first sense. When refining the sense, the system starts by assuming the first sense of the chosen word was the correct sense. This means that even though the system chooses *tattoo#n#3* for *ink*, Refine sense resets it to *tattoo#n#1* until evidence shows that *tattoo#n#3* is more deserving.

The system then performs a mini overlap, similar to the one above, limiting to just the senses of the chosen lemma and their glosses. If a sense other than the first one, had more words similar to the new term than the first one, then it would become the chosen sense of the word. The chosen sense is then cemented as the correct sense and the system moves on to merging or attaching.

2.4 Merge or Attach

The smallest amount of time (in developing this program) was spent on the problem of merging or attaching. This was due to the limited amount of time, and that time being focused more on determining the correct word over whether it should be merged or attached. Our system determines whether the term should be merged or attached by looking at the frequency of the chosen sense as obtained from the WordNet frequency() function. If the frequency was low (if it was equal to zero), then it was assumed to be a rarer sense so the program would attach the new term. If it was higher (greater than zero), then the opposite was assumed and merge was chosen. Our test data results are shown in the following contigency table.

		system		
		merge	attach	
key	merge	7	25	32
	attach	121	447	568
		128	472	600

ink was chosen to be attached to *tattoo#n#3* which means the frequency was greater than zero.

3 Results

On the 600 word test data set that was provided for SemEval Task 14, our system (*UMNDuluth Sys 1*) scored as shown in Table 1.

The SemEval14 organizers also included a baseline score on the data set, which is in the table under *baseline*. As mentioned in the methods section,

System	Wu & Palmer	Lemma Match	Recall	F1
UMNDuluth Sys 1 (2 bonus)	0.3395	0.0984	0.9983	0.5067
UMNDuluth Sys 2 (10 bonus)	0.3857	0.1467	1	0.5567
UMNDuluth Sys 3 (25 bonus)	0.3802	0.2117	1	0.5509
UMNDuluth Sys 4 (50 bonus)	0.3809	0.2100	1	0.5517
UMNDuluth Sys 5 (100 bonus)	0.3735	0.0517	1	0.5439
UMNDuluth Sys 6 (500 bonus)	0.3791	0.2083	1	0.5498
Baseline: First word, first sense	0.5139	0.415	1	0.6789
Baseline: Random synset	0.2269	0	1	0.3699
Median of Task14 Systems				0.5900

Table 1: SemEval Task 14 Scores

originally, our system weighted definitions that overlapped with senses' words as 10 times the length of the word. This was changed close to the end of development as it looked as it might give compound words too high of a score to reach with non-compound words. The 2 times amount was what was submitted and scored above. However, the 10 times amount was run against the same data and scored labeled by *UMNDuluth Sys 2*.

The Wu & Palmer Similarity, as defined by SemEval16 Task 14 task organizers[2], is calculated by finding the "similarity between the synset locations where the correct integration would be and where the system has placed the synset." This score is between 0 and 1. The Lemma Match, again defined by the task organizers, is scored by, "the percentage of answers where the operation is correct and the correct and system-provided synsets share a lemma." Recall refers to the percentage of lemmas attempted by the system. If 600 were attempted out of 600, then recall equals one.

4 Discussion

As the system was being built, we had the idea to originally use more information from Wiktionary[3]. However, when calls to Wiktionary were added in the system, the system slowed down to a halt, taking sometimes over 5 minutes per new lemma, even with the pre-processing. Since it was very impractical to wait this long, additional calls to Wiktionary were taken out of the program.

Another functionality that was thought to be used was the idea of adding in the level of the word in WordNet to the calculations. The level of each word in WordNet was thought to be used in the calculation of merge/attach. Unfortunately, the calls for this information to WordNet slowed the system to a halt at times, this meant the same fate as extra Wiktionary calls.

As seen in Table 1, our submitted system had a recall of less than one. This error most likely occurred because of time constraints. Since the system had to process 600 words, the 600 word file was split four ways to allow four instances of the system to process the data at once. In the splitting of the word file, a word was most likely lost.

Time constraints also was the reason of the difference between UMNDuluth Sys 1 and Sys 2. As stated in the results, Sys 1 had a bonus overlap multiplier of two while Sys 2 had a bonus of 10. The two multiplier was tested only on a small set of words, and little to no difference appeared between Sys 1 and Sys 2. However, it was discovered after the test data was turned in that the bonus multiplier scores higher when it is set to 10 as it is in Sys 2.

When seeing the improvement that occurred between Sys 1 and Sys 2, more tests were run by increasing the bonus. These are shown in Sys 3-6, which appear to peak between the 25 and 50 multiplier.

In the future, it would be interesting to see how additional Wiktionary data could help improve the choice of the system.

[2]http://alt.qcri.org/semeval2016/task14

[3]`http://search.cpan.org/~clbecker/Wiktionary-Parser-0.11/README.pod`

References

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 805–810, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.