Molly Higgins (mfh66)
Sean McNeil (srm274)
Project 2 Report

Process

Parsing Dictionary.xml

To begin our project, we first had to pre-process Dictionary.xml to get it in a format that is easy to parse. We did the following from piazza post 192:
Replace **"** with **'**
Replace **([a-z]*)='([a-z0-9 _%:]*)'** with **\1="\2"**
Replace **gloss='** with **gloss="**
Replace **'/>** with **"/>**

Parsing test-data.data

Followed the same steps as above with replacing certain characters in a text file. We replace the different tenses of the word contained in the <head> tag with the word in the id so that we could easily find the index of the word later on. At first, we overlooked that words contained in <head> tags could be in a different tense than the words in the id. This improved our score. We alternatively considered taking the “stem” of the <head> word, but this risks there being different versions of the word in more place. We also, later on, discovered that stemming was not working the way that we had hoped- another reason to not use stemming here.

Best Sense

For getting the best sense of the each word, we wrote the function best_sense(). This function returns a dictionary of the best senses for the target word and their respective scores. It originally did this by comparing the stem of all of the words in all of the senses of the target word to the stem of all of the words in all of the senses of the target word. As discussed later, we ended up removing this feature because it decreased performance.

The way best sense worked was by iterating over the senses of both words and matching their definitions to each other using semantic similarity for each of the words, overlap, and consecutive overlap. We awarded extra points for definitions that had two matching words one after another, and summed up the score as the consecutive overlap points + the number of overlapping words in general + the semantic similarity of the words using Wordnet’s similarity function.

One thing that we considered doing was simply checking to see if the contextual words themselves were in the senses of the ambiguous word and if the ambiguous word is in any of the senses of the context words. We decided not to implement this technique because the chances of this overlap are very low and not worth the performance time.

Wordnet

In our original strategy, we would get the senses from the dictionary and if they did not exist, then we would go to wordnet. This didn’t use all of Wordnet’s functionality, so we changed our code to go straight to wordnet and then later use overlapping words in the definition in order to map from wordnet to dictionary.xml. We used Wordnet’s similarity functions to check the semantic similarity between words in the definition, and used that score to pick the correct definition.

The weakness of our implementation was this mapping between wordnet and dictionary.xml. There were many cases where we were unable to map the WordNet sense to the senses in our dictionary. This was because we just used overlap to find the similarities between the senses, but this often led to there being multiple correct senses chosen from dictionary.xml. In the case of multiple best senses, we chose one at random. We tried to improve this performance by also getting all the synonyms of the dictionary sense and comparing that to all of the synonyms of the wordnet sense to see if we could map better with the synonym overlap instead. This actually performed worse than choosing at random so we changed it back.

This issue arises mainly to because of the same roadblock we faced early on when trying to complete our implementation: when just checking similarity using word overlap, the results are extremely poor. This makes both our results and our mapping very poor.

When using wordnet for mapping, we return all the definitions of all of the synonyms of the word. We made this choice because our dictionary is very small and it is very likely that the context words are not in our dictionary.

One thing we would use in the future is Wordnet's lemmatizer feature, but as discussed in the results it would not make sense to use that due to the drawbacks of our implementation.

Senses

Each sense will get a score for the number of words that are in the description of the sense that the target word sense is being compared to. One drawback of calculating score in this way is that senses with longer sense definitions are more likely to contain words that appear in descriptions of the compared sense. At first, when comparing the overlap in definitions, we compared the senses of the target word to itself. This resulted in the longest sense definition always being selected and our project received a very low score. We fixed the comparison.

This mistake however, made us think about another feature that we considered implementing: some sort of factor that accounts for how long the description of the sense is. For example, if we have an extremely short, but accurate sense description such as "mouse: a small rodent" and then a very long description of another sense like "mouse: a small handheld device that is dragged across a flat surface to move the cursor on a computer screen, typically having buttons that are pressed to control functions." The overlap is much more likely to overlap with the definition that is longer. In the shorter definition of mouse, there are only three opportunities to overlap. If we could somehow weigh the chances of overlap, we may have been able to receive a higher score.

Stemming

We tried to stem all of the words in the sense of all of the context word. We chose to do this so that we would get overlap between words like "running" and "run" in the definition. However, in our tests, we removed this and it performed better without stemming(). This result is still very confusing for us- however, we think it is most likely because our inaccuracy lies in the mapping from wordnet to dictionary.xml, and just by strange circumstance and perhaps luck removing this stemming causes our matches to increase by .1%.

Lowercase

We also chose to convert all of the words in all of the senses to lowercase so that “He” and “he” will both count as overlap

N Context words

We compare our target word with N words to the left of the target word and N words to the right of the target word. We would expect that having more context words would result in higher accuracy, but would take much longer to complete. Eventually, increasing N no longer affects the accuracy of the sense disambiguation. This is because context words become irrelevant after enough context words are known, and after the context words are so far away from the target word that they are not really context words at all anymore.

One possible feature we thought about implementing was a way to take into consideration how far away the context word is from the target word. If the context word is directly before the context word, it could mean more a word further away from it. However, we do not know how true this is and it may be very inconsistently that closer words matter more. It was not worth our time to implement it.

Interestingly enough, when we experimented with changing N, our results did not change much. One possible reason for this is that our mapping between the given dictionary and WordNet’s dictionary is not very good. If we do not know the correct corresponding dictionary sense to Wordnet’s sense, we cannot improve our score for that unknown sense. This is discussed in our results.

Consecutive Overlap

The sense will get an additional weight added to the score if the words have consecutive overlap. We tried adding weights of 0.5, 1, 2, and 5 for each consecutive overlap. We found that adding 2 was the best, but by marginal amounts.

The marginal differences in the reward for consecutive overlap could be explained by the inconsistency in the value of the overlap. In some cases, consecutive overlap could be very valuable, like in the case of “pine” and “cone”. However, there are many other times when consecutive overlap would reward overlap in words that are not meaningful. For example, “this” followed by “is” is a common phrase that would receive extra weight for consecutive overlap but has very little contextual meaning in reality. This problem could be helped by removing stop words.

Multiple Best Senses

Originally, we attempted to account for words having two best senses. We did this by checking to see if two senses had the same score. If they did, both senses are added to the best sense array. We later decided to remove this feature as less than 10% of the data set have more than one best sense. Our sense scores were also not accurate enough to make this worth including.

Stop Words

We did not implement stop words, but this may have been very valuable rewarding overlap and efficiency. If we had removed stop words before rewarding overlap, we would not give higher scores to senses that have similar meaningless words and would not have rewarded meaningless consecutive overlap such as “this is” as described above. With this being said, we did not implement this feature because there was already such little overlap between senses and we really needed to focus our attention on the larger issues instead.

Removing stop words would also increase efficiency because it would mean that we would not have to look up the senses of a meaningless word such as “of” or “this”.

Part of speech tagging

We considered using part of speech tagging, but we were not sure how part of speech tagging would benefit us in a dictionary based word disambiguation. One possible way part of speech tagging could have helped us in efficiency is if we checked to see if the word was a noun or verb rather than a preposition or article and only checked the nouns of verbs. The closest thing we did to part of speech tagging was removing the words “and,” “the,” “it,” and any words not in dictionary.xml from the definitions before moving on to find the similarity between the two words. This eliminated over rewarding definitions that had a lot of prepositions or other common words that wouldn’t benefit the score matching. In this same part, we removed all the punctuation from the test sets to avoid over inflation because of matching commas.

Software

We imported stemming and xml.ElementTree to help with our code. We did not end up using stemming because it made our results worse somehow. We used xml.ElementTree to parse the data.

Results

At first, when comparing the overlap in definitions, we compared the senses of the target word to itself. Because of that, our function always returned the longest sense definition as the best sense. This gave us worse than baseline testing with a score of 0 on Kaggle.
To figure out if our error was in the format of our file, or if we were actually doing worse than baseline, we decided to write code to actually calculate the score of the baseline and see if we could get a score with the same format. With this we simply chose the first sense of every ambiguous sense. This gave us a score of 0.16734 on Kaggle. We then were able to focus our attention on why our score was so low. One of the major issues that we faced was that the words were not getting enough overlap. To solve this, we tried to analyze all of our words using WordNet and then mapping the words back to WordNet. This improved our score slightly to 0.17901 on Kaggle. It was still so low because we had trouble mapping back to WordNet. We tried to improve our score with adjustments in several parts of the code. These adjustments and corresponding scores are recorded in the data below:

Note* Our scoring does not take into account instances with more than one correct answer. It will simply mark the answer as wrong, therefore how testing should be slightly harsher than the actual upload.

Description of what changed	Our Score
If more than one sense matches wordnet definition, choose a sense randomly, includes rewarding consecutive overlap, stemming, lowercase, rewards consecutive overlap with +1, N = 1	0.11983
If more than one sense matches wordnet definition, choose a sense	0.18318

randomly, includes rewarding consecutive overlap, **no stemming,** lowercase, rewards consecutive overlap with +1, N = 1	
If more than one sense matches wordnet definition, choose a sense randomly, includes rewarding consecutive overlap, **no stemming,** lowercase, **rewards consecutive overlap with +2,** N =1	0.18560
If more than one sense matches wordnet definition, choose a sense randomly, includes rewarding consecutive overlap, **no stemming,** lowercase, **rewards consecutive overlap with +0.5,** N = 1	0.18407
if more than one sense matches wordnet definition, check to see if the synonyms of the dictionary word match the wordnet synonyms and map using word with best score, no stemming, lowercase, **rewards consecutive overlap with +5,** N = 1	0.18954
if more than one sense matches wordnet definition, check to see if the synonyms of the dictionary word match the wordnet synonyms and map using word with best score, no stemming, lowercase, rewards consecutive overlap with +2, N = 1	0.18954
if more than one sense matches wordnet definition, check to see if the synonyms of the dictionary word match the wordnet synonyms and map using word with best score, no stemming, lowercase, rewards consecutive overlap with +2, **N = 4**	0.18954
if more than one sense matches wordnet definition, check to see if the synonyms of the dictionary word match the wordnet synonyms and map using word with best score, no stemming, lowercase, rewards consecutive overlap with +2, **N = 8**	0.18954

Interestingly enough, when we ran the best results of these in Kaggle - using “if more than one sense matches wordnet definition, check to see if the synonyms of the dictionary word match the wordnet synonyms and map using word with best score, **no stemming**, lowercase, rewards consecutive overlap with +2, **N = 4”,** our score went down to 0.16838 on Kaggle. One possible reason for this is that the issue lies in our mapping from our dictionary senses to WordNet senses and the inaccuracy in that is so high that these little tweaks are making negligible differences.

Discussion

Our project’s results and process were very surprising to us. Hitting 0% accuracy was a huge roadblock for us, and we could not figure out for a while how it could be mathematically possible to get so low of a score. Our original takeaway was that the lesk implementation was not a good way to go, but when looking at statistics for this algorithm it is supposed to produce 40%-60% accuracy, so we concluded we were missing something fundamental in implementing it. It was the semantic similarity functions from wordnet that even got us actual results on the table, and from there we hit the same problem again: mapping two senses together only using overlap. From there, there was not much we could do to raise our score without finding a new strategy that was neither semantic similarity not word overlap, and at that point we had already drained most of our time and resources trying to get the former strategies to produce results.

The nature of our solution made it difficult to try and improve our results using WordNet's features or other strategies, since we had mostly hit a brick wall trying to use overlapping words. We feel like this roadblock forced us to find other methods to try and succeed, but made it difficult to improve our score with our workarounds. While there were many methods that we thought about implementing, and understood their importance, they would not be useful without having successful mapping between our dictionary senses and WordNet's.

We are very curious to see how other strategies worked better after submitting our assignment, since we had so much difficulty even hitting 20% of the matches. We were very surprised that the naive dictionary mapping implementation did not even work at all, so we could not build up naturally from there without revamping our entire project and starting mostly from scratch. One thing we would do in the future is try and match the senseids of the definitions from wordnet to the dictionary.xml definitions that came from wordnet.