**Assignment2 Report**

hp343

jz578

sg889

yt438

Table of Content

1. Approach 2

1.1. Dictionary-based WSD 2

1.2. Extension 2 4

1.2.1. Dictionary-based WSD 4

1.3. Extension 3 4

2. Software 4

2.1.1. Dictionary-based WSD 4

# Approach

## Dictionary-based WSD

* Preprocessing

Convert dictionary.xml into this data structure:

|  |
| --- |
| dictionary [  "begin.v":{  num:1  senses:[  {  id:1  wn\_ids:[1,2,3,4]  gloss:["hello","world"]  examples:[  ["hello","world"],  ["how","are","you"]  ]  },  ...  ]  },  ...  ] |

During the conversion, we did a small adjustment to dictionary.xml, and named it dictionary-modified.xml. This modification aims to solve the quotation mark conflict in python’s xml parser. This adjustment does not interfere the accuracy of result.

And we convert validation.data into this data structure:

|  |
| --- |
| validation\_list [  {  word:"begin.v"  real\_sense:1  target\_word\_idx:15  sentence:['pressure', 'hello']  },  ...  ] |

During the preprocessing, we do the stemming and lemmatizing. Besides, we also remove stop words, for example “a”, “an”, we use a stop word dictionary named “Lucence” to remove all stop word in the word context, examples and gloss.

* Calculating

1. hit count

Count how many same words are there in the word context and in a sense’s gloss and examples.

2. Consecutive count

Count how many consecutive words are there in the word context and a sense’s gloss and examples. If there are two consecutive words, for example, in word context [“ab”, “cd”, “ef”] and in an example, [“ab”, “cd”, “gh”], then we add two to the consecutive count.

We add the hit count and consecutive count together, and return the sense with max sum as a best match.

* Scoring

If the best match sense calculated is the right one, then we add one to match count. After checking all the items, we divide match count by the total item number, and we get a score ranging from 0 to 1, inclusive.

* Screenshot of Kaggle

## Extension 2

### Dictionary-based WSD

We simply choose the second largest sum of hit count and consecutive count. And we change the hit rule, if the real sense is in one of , it goes up to

## Extension 3

We redesign the score system, if the best sense matches, we add 0.7 to the match count and add 0.3 if a second best sense matches. After checking all the items, we divide the match count by 0.7 \* total item number. We will get a score ranging from 0 to 1, inclusive.

This score makes more sense because it evaluates the overall performance of our model, not only the best match result derived from the model. The second best match can also achieve some score.

# Software

We use the LancasterStemmer and WordNetLemmatizer library from nltk to do stemming and lemmatizing.

### Dictionary-based WSD

Change the **glob\_valid\_path** and **glob\_dict\_path** in dic\_preprocessing.py, so that these two variables contains the address of validation\_data.data (or test.data)and dictionary-modified.xml

If you want to run the basic version, just uncomment the line “**print basic\_score(dictionary, validation\_list)**”, and run “**dic\_wsd.py**” in command line tools.

If you want to run the extension 2 for dictionary-based WSD, uncomment the line “**print second\_best\_match(dictionary, validation\_list)**”, and run “**dic\_wsd.py**” in command line tools.

If you want to run the extension 3 for dictionary-based WSD, uncomment the line “**new\_score\_match(dictionary, validation\_list)**”, and run “**dic\_wsd.py**” in command line tools.

# Results

## Dictionary-based WSD

* Use stemming

As mentioned before, we do Lemmatizing does not change the word quite much. However, when it comes to stem, things are quite different. For example, in the word “begin.v”, in the examples, there are past tense, “began”, however, after stemming, it turns out to be “beg”, and “begin” remains to be “begin”. This is not we want, what we want is just convert “began” to “begin”. So we decided to remove stemming at first, however, after calculating the hit rate on validation.data, we found out that after stemming, the rates goes higher (score goes from 0.424437299035 to). So eventually, we add the stemming process into our program.

* Sum up hit count and consecutive count

This is not our first design, we used to take the max length of consecutive words as consecutive count, and it comes with highest priority. This means no matter how big the number of hit count of a sense is, if it have a very low consecutive count, it won’t be the best match. The basic assumption is that if we have a comparatively long consecutive sentence, it’s more likely to be the best match sense because even other sense has more hit count, the hit might be caused by very popular words. However, this approach does not work as well as we expected. After checking the result, we find out that the max consecutive word length is relatively small, usually 1-2, this may not be very distinguishable. Besides, we remove most stop words before, which makes it harder to find consecutive match words. This method achieves a basic score of 0.488745980707

# Discussion

Dictionary focuses on analyzing the similarities between word context and different senses. So it more suits the situation where the context of a word is long enough so we can extract enough information from that.