CS372 HW3 report

20180368 An Haechan

**1. Explain Algortihm**

My algorithm consists of 3 tasks: 0-Raw data into tagged corpus 1-Extract heteronyms existing in the corpus, 2-Annotate pronunciations of them, 3-Rank them.

**0. Raw data into tagged corpus**

I used comments in r/wordplay of reddit for the corpus, since it has multiple occurrence of heteronyms but not too much. And got raw data using Convokit api.

Then tokenized and tagged using nltk tokenizer and tagger.

**1.Extract heteronyms**

At first, I constructed the vocabulary of corpus, which is set of lemmatized(by wordnet Lemmatizer) words appeared in the corpus. (Used lemmatizing in order to reduce crawling process)

And then crawled the data:(POS, pronunciation) of each word from the Cambridge Online Dictionary.

For every word in vocabulary, if there exist two or more pronunciations for it, I marked it as heteronym.

Since there was no clear criteria of ‘how much pronunciation &meaning should differ to be a heteronym’ in the HW description, I chose to make a word heteronym even if there is only small difference in accent and meaning. ‘Different IPA string’ means different pronunciations. And judging by the fact that ‘rebel’ (rebel(N) just means ‘person who rebel(V)’) is defined as heteronym in the HW description, I assumed if pronunciation is different so described in different section in since Cambridge, then meaning will be different. (In Cambridge dictionary, little pronunciation difference but with the same meaning is in the same section, so my assumption is valid.)

I excluded abbreviation.

After extracting heteronyms, I crawled from the dictionary again to get ‘definitions’ of each heteronym. There exist one or more definitions for each heteronym.

**2.Annotate pronunciations**

First, using the extracted heteroynms, find the sentence contains the heteronyms.

In this process, I used lemmatization to the words in sentence to find cases where -s, -ing,… follwos.

However, I didn’t use the lemmatized word, if the word in sentence was past participle or past, as in such cases the pronunciation changes irregularly like do to did. But still there is also cases where only -ed or -d adds at the end, I considered this case by not lemmatizing, just split the ed or d at the end and see if the result is in heteronyms.

Input is word ,POS and sentence, and Output is pronunciation.

There is two cases: (1) There exist one or more POS which has two or more pronunciations (2) Otherwise

For case (2), it is a straightforward process to choose pronunciation. First lemmatize the word and search the word from heteronym dictionary, then choose pronunciation correspond to POS.

For case (1), since the ML is banned in this HW, I used dictionary-based method for word sense disambiguation using wordnet and lesk. First, find the expected synset of word in the sentence, using lesk(gave tag as input to increase accuracy). And then, there are list of ‘possible pronunciations’ for that word. As I mentioned earlier, we have ‘definitions’ data for each pronunciation. For each pronunciation, for each definition, find the ‘first word in the definition that has the same tag as desired word(heteronym of which pronunciation is what we’re finding)’s tag’. I used this trick because generally, in the definition, similar word is usually used in the front. With this first word, get synset of it and compute path similarity with desired word’s synset. Get maximum value of path similarities (from definitions) and set it as ‘path similarity of the pronunciation’. Among all possible pronunciations, choose one with highest similarity.

**3. Ranking**

I interpreted the ranking as following:

1. Number of heteronyms /2. max ‘portion’ of heteronym

Portion(heteonym) = (number of heteronym) / (number of heteronyms)

For heteronym in heteronyms, find max portion(heteronym)

3. negative of (number of ‘POS-unique’ heteronyms)

Heteronym is ‘POS-unique’ heteronym if there exist only one pronunciation for a POS,

**2. Discuss result & Further improvement**

Since reddit is an internet community with casual atmosphere, there were some very long sentence without period, including one which is a ‘code’ of programming language. Due to the fact that do, use are clearly heteronyms, according to the definition, most of the high-ranked sentences were long sentences with do and use. I guess it is quite different from what intended, (distinguish hard heteronyms like wind and tear in the same sentence) but it makes complete sense for our definitions. For further improvement, it could be better to revise the ranking, first priority to ‘ratio of heteronyms out of entire words in the sentence’, to exclude the ‘very-long but normal sentence’.

Also, there was a sentence where do is used as onomatopoeia, like “do doo do doo, what is this song?”. Because the Cambridge dictionary doesn’t count this cases, it was hard to remove such case. I thought about if certain word repeats contiguously, then removing it but the corpus was ‘wordplay’ so there could be many other sentences with repeating words.

Because majority of heteronyms are ‘POS-unique’ heteronyms, heteronyms that are not ‘POS-unique’, such as wind, tear did not appeared many. So it’s a bit hard to judge about my algorithm from the result data. But if you see the whole data, there exist some of them, like row, are correctly annotated.

There were some cases that the word is tagged with wrong POS, so the pronunciation annotations couldn’t work since we first list ‘possible pronunciations’ by the POS. In this case, I just tagged with the first pronunciation in the value of dictionary[word]. But if I use ‘most frequently used tag’ from the corpus, it would be better. Also more basically improvement in tagger will be helpful.

I have another idea to increase accuracy of annotation, that is ‘using the length of syllables of the sentence.’ First, from the vast corpus, collect the frequency distribution of ‘the total length of syllables in the sentence’. And then, when annotating heteronym with different number of syllables, choose the syllable that makes the entire length of syllables of the sentence more probable one(in the frequency distribution). I think we can use this criteria, combined with existing method. But how exactly is undefined, yet, just an idea for further improvement.