CSC485 A2 Writeup

Richard Yan

October 2022

Family name: Yan
Given name: Richard
Student #: 1005731193
Date: Nov 4, 2022

I declare that this assignment, both my paper and electronic submissions, is my own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters and the Code of Student Conduct.

Signature: [illegible]

0. Warming up with WordNet and NLTK

(a) The synset is Synset('rock_hind.n.01'), and it has 2 paths:

Path: [Synset('entity.n.01'), Synset('physical_entity.n.01'), Synset('object.n.01'), Synset('whole.n.02'), Synset('living_thing.n.01'), Synset('organism.n.01'), Synset('animal.n.01'), Synset('chordate.n.01'), Synset('vertebrate.n.01'), Synset('aquatic_vertebrate.n.01'), Synset('fish.n.01'), Synset('food_fish.n.01'), Synset('sea_bass.n.02'), Synset('grouper.n.02'), Synset('hind.n.01'), Synset('rock_hind.n.01')], depth: 16

Path: [Synset('entity.n.01'), Synset('physical_entity.n.01'), Synset('object.n.01'), Synset('whole.n.02'), Synset('living_thing.n.01'), Synset('organism.n.01'), Synset('animal.n.01'), Synset('chordate.n.01'), Synset('vertebrate.n.01'), Synset('aquatic_vertebrate.n.01'), Synset('fish.n.01'), Synset('bony_fish.n.01'), Synset('teleost_fish.n.01'), Synset('spiny-finned_fish.n.01'), Synset('percoid_fish.n.01'), Synset('serranid_fish.n.01'), Synset('sea_bass.n.02'), Synset('grouper.n.02'), Synset('hind.n.01'), Synset('rock_hind.n.01')], depth: 20

1 The Lesk algorithm & word2vec

(d) This extension is helpful because that without the extension, the overlapping size might be determined by the signatures (i.e. the definition and examples of that single sense is just too small), but with the extension, it's more likely to be determined by the context (i.e. we have a large signature compare to the context now), so if they are the same sense, it's more likely to have overlap appear in some other synsets definition and example related to that sense.

(g) One-sided Lesk's performs better than just the cosine, and I believe that's because we put more focus on the context (in which the word we want to disambiguate is in) rather than both the context and the signature. For example if we are comparing signature = new,buffalo,york and context = buffalo,buffalo,like, then just the cosine similarity will give us approximately 0.516, while the one-sided will give us approximately 0.894. The idea is, the words in the signature might simply just be too much to consider (i.e. too many irrelevant words that makes the normalized vector too sparse).

(h) The dot product of the numerator will simply become number of words that appear in both context and signature. The denominator will become proportional to the product of numbers of distinct words in context and signature. This is like doing a set intersection and count how many distinct words are there in the intersection, and normalized it.

(i) mfs: 41.6%
lesk: 39.4%

lesk_ext: 45.8%
lesk_cos: 38.1%
lesk_cos_onesided: 43.4%
lesk_w2v: 47.4%

(j) mfs remain unaffected, lesk became less accurate, lesk_ext became 0.1% better, lesk_cos became less accurate, lesk_cos_onesided became 0.1% lower, and lesk_w2v became slightly (0.3%) lower.
I believe this is because now some upper case word that isn't stopword will now be treated same as their lower case word. Since sometime the word are in different sense because they are in the beginning of the sentence (i.e. some senses are more likely to appear at the beginning of the sentence, etc.), hence affecting the simple overlap algorithm (lesk) the most, while extended lesk has huge signature hence maybe able to weaken the affect of that. Cosine similarity is also affected probably because having a large signature the, the two vectors are having too much vector dimensions hence for all senses of that word the cosine similarity might be pretty close, hence some small change (upper to lower case) will affect some predictions. Onesided cosine and w2v pretty much stay unaffected because one use only words in context to build the vector (hence less affect since probably there is only one upper case word of that single sentence context), and w2v itself use lower case to find vectors as the second best choice.

2 Word sense disambiguation with BERT

(a) I believe context is necessary. One example I can think of is 'Call me Tim.' and 'Call me, Tim.' It's clear that the word 'Call' here represent different meaning (hence different sense), but in those methods in Q1, they can never be able to disambiguate this, because they didn't even look at punctuation. I think that punctuation is also sometime providing a decent amount of information towards the sense of a word, and we should take account of that. In particular, there is another example, which are the sentences 'I watch the show' and 'Show me the watch'. Methods in Q1 can't detect this since I and me are in the stopwords I believe, hence they will predict the same sense for watch/show in both sentence. The more general pattern is that, those homographs sometime can have totally different meanings while having approximately same words in the sentence, so detecting order will also be very important to disambiguate word senses.

(c) After sorting, the length of the sentences in each batch will be approximately the same, hence resulting that length of the final tokenized sentences are also close enough, which will reduce the amount of looping on padding, hence reduced the overall runtime. (i.e. we can see that the first batch after sorting will only contain sentence of length 1 and 2, while length of tokenized sentence only get up to 6. But if we have a length 10 sentence in it, every tokenized sentence will be padded up to maybe 15-20, which will be very inefficient).

(e) If we use our algorithm to disambiguate arbitrary sentences, one issue will be that it's much more time consuming, since according to our algorithms, they will need to explicitly disambiguate each of the word at the same time, for example in the Bert_based algorithm in q2, if we don't know which words to disambiguate, then we have to explicitly calculate the best sense of every sense of every ambiguous words.