NLP Homework 8

Team Members , Paul May and only Paul May

Team contributions, all parts done by Paul May

The following is a short essay done at the suggestion of Vrindavan, who noted that if we beleive our F1 score does not properly represent the effort interned in the project we should write a few hundred words about it. I have made many very experimental attempts to improve my F score, most of which were very novel and time consuming. I believe I deserve more credit than my F score represents as is demonstrated by my spectacular and very time consuming (tens of hours) failures. I believe that this time commitment consitutes a 'good faith effort' as noted in the rubric and I hope that my efforts , even if fruitless , entertain you a little.

First Failure: Pronoun Resolution

I attempted to design a very simplistic and fast pronoun resolution algorithm and metvaried sucess. I estimate it correctly fills in about %60 of pronouns with the relevent anaphora. I used this to increase overlap in the mc500 data question keywords and sentence keywords to form a better baseline. The improvement was about %3 overall and resulted in much more technical debt and time consumption than it was worth competitively.

Anaphora Algorithm Process

1. convert text to a long pos tagged chain of words, retag title cased words that do not occur at the front of sentences or are obviously not mistagged names as proper nouns.
2. make a dict containing the the relative frequency of each proper noun in the document. When parsing this I used a special chunker and grammar such that lists of names would register as plural and never be interfere with singular proper nouns.[1]

a. Chunker chunked noun phrases in the form:

NounPhrase ('and' or ,) NourPhrase (('and' or ,) NounPhrase)* [2]

the exact grammar is in utils.py

3. move through the sentence chronologically building a list of possible anaphora for the next pronoun. when the pronoun is encountered check a dict of all previous decisions regarding the gender, plurality, or inanimate nature of the noun.[3]

4. when the most likely pronoun has been chosen , i.e. the one with the highest saliency score. select it, update the relative frequencies, and then keep parsing.[4]

The algorithm itself only managed to imrpove overlap in who questions by about 3% , which, considering that it is an actual across the board improvement , is not actually a poor result. it is just far less than would be indicated by the amount time spent concocting such an elaborate and semi-broken thing.

Second Failure : Wordnet Similarity for Baseline Improvement

in a misguided attempt to improve baseline , which i'm starting to think is a bigger problem than this project[5] i attempted to make a modifier for my metric based on the similarity of synsets found in the question and sentence keywords. this attempt can be found in utils.py under get_similarity. in theory it should have worked, in practice it cut recall about 10% and consumed alot of work hours.

wordnet baseline attempt process

1. find question keyword wordsense in the csv files , if it isn't there and the synsets list length is one , select that synset. if the synset list length is two, select the first synset. otherwise move on to the next keyword

2. once the synset list has been constructed repeat for the sentence being analyzed. producing another list of synsets

3. compare each question keyword synset to each sentence word synset, returning none if either are empty lists, returning the highest path similarities other wise.

4. add all similarity scores. this should , in theory if all synset senses were captured properly. result in a list where word matches give a 1. strongly synonymous words give a 1 aswell. and light matches give a float value. this is not what happened

Due to some bug in my code that whas either not worth workign out , or some design flaw that was not worth redisgning for a whole new try. this method which i thought might have introduced some new positive noise to the sentence selection method was made ineffective and useless. I have not removed it from the code entirely of posterity sake but it was a massive waste of time in search of a heuristic that turn on and off cases by case to improve sentence retrieval accuracy. unfortunately for NO question type does it improve accuracy.

conclusion:

between these two tasks I wasted probably 48 hours between four days of work. this may sound like an exaggeration but I really have been putting time into this. I like novel approaches , and I find myself obsessing over finding them when I am confronted with problems like these. I know there is a novel way to achieve a better baseline without using Google word vectors. I wanted desperately to avoid word vectors for a few reasons, namely file size , run speed , but more over deep control over my data set. i'm not a large fan of wordnet for this reason as well. wordsense disambiguation is not a well solved problem , and until there is an off the shelf method that performs with high accuracy i think i can safely say wordnet similarity is useful for the most part in niche cases.

While I knew I was potentially sabotaging my grade and sabotaging my position in the competition I acted motivated by exploration in an open and unsolved field. even if what i did was stupid , maybe the field just needed someone as stupid as I am to find a novel method. in my opinion the work I have demonstrated put into this project shows a good faith effort, an act which coaligns with the spirit of the class even if it may seem a little nescient and misguided.

thank you for your grading discretion.

Footnotes:

[1] Note on iteration structure: after encountering a group it is necesary to consumer a number of loops equal to it's length in words as to not include any of the sub items as an individual in the pronoun resolution data structure.

[2] Note on grammar: this structure only sold because of the simplistic nature of the fables which I was parsing, I also excluded non-proper nouns from the noun phrase definition in order to reduce accidental inclusion of arbitrary elements in the group when it was included in later anaphora replacement.

[3] Note algorithm results: this has the nasty side effect of making poor decisions last, with further tweaking to this mechanic and the gender dict I probably could have improved the results of the algorithm but I simply ran out of time to fuddle with soemthing so useless.

[4] Note on saliency scores: I did not follow the example weights found in the Leass and Lappin algorithm. their use case was a fair bit more complex than mine and probably could not have used total document frequency as a good indicator of probable pronouns.

[5] Note on baseline: Most improvements to the baseline come at signifigant time cost to the programmer and are far too case specific to generalize well. improving keyword overlap and keyword spiking seems to be the best way to gain a few points here and there. but at the end of the day question type sparseness and the overwhelming presence of 'what' questions makes these improvements marginal at best. improvements to 'what' questions are also marginal because at best you can imrpove one third of the questions at a time as the return type and query keywords are either verb or noun oriented and ambiguous by nature.