Research Project

Building a Natural Language Opinion Search Engine

Natural Language Processing

Instructor: Arjun Mukherjee

1. Background

We learnt about Boolean text retrieval in class. Please refer to slides/lecture notes given in class and algorithm details in these drafts (with links) [Schutze, 2014], [IIR Chap 01]. We now want to perform a practical task where you want to search for relevant and coherent opinions. This would require you to "teach language" to a machine in a way that it *understands natural language* and not just perform queries algorithmically.

2. Corpus (document collection)

You are provided with a real world review corpus from Amazon.com which contains a motley collection of reviews of electronic and software products. You also have various metadata associated with the review such as the review title, the star rating, the product and customer id, helpfulness votes (helpful_count), total votes (out_of_helpful_count), # of comments, etc. The SQL dump of the data containing all metadata and detailed schema is provided in reviews_segment.sql available at this location: http://www2.cs.uh.edu/~arjun/courses/nlp_ugrad/hw_proj/res_proj/res_proj.7z

OR

http://www2.cs.uh.edu/~arjun/courses/nlp_ugrad/hw_proj/res_proj/res_proj.zip

The resource also contains the raw dump of review text and 50 major topics in the corpus to get you a feel of the dataset.

Bootstrapping: If you prefer to work with flat files/other format suitable to you, you can transform the data in the appropriate format. E.g., just read the .sql as text in Notepad++ and then parse the rows as entries in your internal format. You can use a standard SQL client (e.g., HeidiSQL) and some SQL Database (e.g., MySQL) to load the contents of the data into your SQL database.

3. Problem definition

Our goal is to perform opinion queries on the data. This is a highly practical situation where you want to buy a product (say on Amazon) and therefore want to first read the reviews mentioning some key feature or aspect of a product and the reviews mentioning the positive and/or negative aspects of that feature. We relax the problem whereby we don't care about a specific product (as that would require you a very large database of all reviews of those product and might become Web scale!) but are only interested in retrieving relevant reviews mentioning an aspect/feature and an opinion. To further simplify matters, we assume all our opinionated queries take the following form $q = [aspect: opinion]$ and aspect and opinion both can be at most two words. Consider the following example queries:

$q_1 = phone\ screen: issues$; $q_2 = wifi\ signal: weak$; $q_3 = mouse\ button: great$; $q_4 = battery\ life: long$; $q_5 = printer\ ink: expensive$

Ideally, when you place these queries, our goal is to retrieve documents (reviews in our case) which talk about the aspect and place it in the relevant opinion orientation/light.

4. High Level Approaches

This is an open ended research problem and clearly there are multiple ways of solving this. The goal is to try and leverage concepts and ideas from all areas of NLP covered in this class. However, you may come up with more neat and novel ideas for a solution and are strongly encouraged to do so in this research project. Below you will find listed some of the nascent ideas in high level versions that can be directly built from the basic concepts learnt in this course and help you in your thinking.

4.0 Preprocessing: We can filter certain words (e.g, those appearing less than 5 times) and some stop words from this list: https://gist.github.com/sebleier/554280 as these words appear much less frequently compared to the bulk of other important words and filtering them can expedite computation. You can also apply lemmatization/stemming to reduce the vocabulary space. However, we need to be careful to not miss out any important content words. Another new wrinkle here is to capture smileys such as :-) and :-(and their variations using regular expressions to indicate a positive/negative opinion placeholder. For progressing fast, you reuse the pre-processing/parsing modules from the mini project (DatasetFormatter.java).

4.1 Baseline (Boolean search): Given that we have built a Boolean search engine already, we can clearly use that as a baseline. Merging aspect and opinion segments of the query and using AND on all terms gives us all documents mentioning those terms. You might also want to OR the aspect terms and AND it with the opinion terms (e.g., for $q = [wifi\ signal: drops]$, we can issue a query: $(wifi\ OR\ signal)\ AND\ (drops)$). This would get you all reviews mentioning some of the aspect and opinion terms thus growing your result set to also include other (possibly) relevant reviews.

4.2 Boolean and Rating Search: You can augment the Boolean search on text by also considering the rating of a review. You can do this with the help of an opinion lexicon (e.g., the lexicon of positive and negative opinion words available here due to [Hu and Liu, KDD, 2004]). One naïve first approach is to look up the opinion in the query as positive/negative and if positive, then we can AND the query terms with reviews having star rating > 3. For negative opinions, we may AND the condition that the star rating is <= 3.

4.3 Modeling linguistic relevance using classification: Another vertical approach is to employ Boolean retrieval using the aspect terms only (you may try using OR/AND in the aspect terms to retrieve more/less documents depending upon what is more suitable). This would yield us all reviews mentioning the aspect terms. The next step is to find the relevant reviews, i.e., reviews mentioning the aspect in positive/negative light. This could be achieved by a classification model (e.g, SVM, Naïve Bayes, etc.). Although this can be done at various levels, we discuss a plausible idea at the review level. We can use a decent portion of reviews (e.g, 70%) for building a positive/negative review classifier and later using the classifier on new queries. The labels can be obtained using the star rating (e.g., reviews with rating > 3 is positive, while reviews with <=3 are negative). We can use words, POS tags, as the features of the review. The rationale is to use this classification model is to classify retrieved reviews containing the aspect terms as overall positive/negative and filtering those reviews which contradict the opinion orientation of the query.

The above method may be not be very good as the overall orientation of the review may not necessarily corroborate with the opinion orientation of the aspect in the query. A more fine-grained alternative is to use a sentence classifier. That would need some more thought (e.g., in obtaining labels for sentences) but is definitely worthwhile. A sentence level classifier also lets you leverage with more language structures (e.g., you can now use the chunks NP, VP, etc. as features) which can be helpful in ascertaining the opinion orientation based relevance in the sentence.

4.4 Grammar and structure based relevance: After having retrieved several documents (reviews) containing the aspect terms, we may further apply some of the following ideas/heuristics for pruning irrelevant reviews:
(a) Using grammar with opinion orientation: These are more fine-grained rubrics of relevance. For instance, are the opinion word and aspect in the same sentence? Or does the aspect contain an opinion word of the same orientation within a window of certain words? Even better is to use a dependency parser and parse trees (e.g, using Stanford Parser) which can provide you the opinion terms modifying specific aspect terms in a sentence. All these techniques can be used to significantly improve your relevance. You might want to try to design a scoring function to rank the relevant reviews/documents (e.g., reviews where a similar opinion is used on the same aspect should be more relevant than a review mentioning the aspect and a general negative opinion on other sentences).
(b) Using review title and Sentence Structure: Review title and its sentiment orientation may be useful in improving relevance on retrieved documents. Another useful heuristic is to use overall positive/negative word ratio in the retrieved reviews/sentences of those reviews where the aspect is mentioned to check whether the review orientation is positive or negative.

4.5 N-gram based filtering for pilot experiments: You may also use an ngram language modeling toolkit (e.g., Kylm, BerkeleyLM, KenLM, SRILM) to find frequent aspect ngrams (bigrams in our case) and focus your experiments on relatively frequent aspect terms/ngrams (e.g., those appearing at least 15 times) as we care about important aspects and not a remote aspects. N-grams may also be used for finding collocations of some typical opinion that is likely to appear with a query opinion (e.g., "ergonomic problem" seems to be a likely bigram in the mouse/keyboard reviews). If your could discover *ergonomic* then you can actually add it to the query to improve relevance.

4.6 Topic models: This might be an advanced concept. But topic models such as Latent Dirichlet Allocation (LDA) [Blei et al., 2003] may also be used to find likely collocations on a random segment of the data. To help you leverage this powerful tool, you are provided with 50 most mentioned topics in the data. Notice that you can add some heuristics from the topics terms in your query to make results more relevant. A topic is a collection of words which all represent a semantic concept.

5. Result Evaluation

Our goal here is to see whether we could discover any meaningful documents (reviews) which are relevant to our query beyond what we can obtain using a baseline. Often times the baseline can yield several results however only a few might be relevant. You will be tested on the relevance of documents retrieved on the following queries only (compute Prec. as # Rel./# Ret. Where Prec. refer to the precision, # Rel. refer s to the number of documents that are relevant and #Ret. refers to the number of documents that were retrieved):

Query	Baseline (Boolean)			Method 1 (M1)			Method 2 (M2)		
	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.	# Ret.	# Rel.	Prec.
audio quality:poor									
wifi signal:strong									
mouse button:click problem									
gps map:useful									
image quality:sharp									

A principled approach is to try and implement different approaches/ideas or some useful combination of the above ideas on the example queries (in Section 3) and other queries you can think/formulate from the data and optimize each method. Then compare different methods/approaches (say you implement 2 different methods M1, M2) and report the performance of each method under 3 metrics: Ret. (total documents/reviews retrieved), Rel. (# of relevant documents retrieved), and Prec. (Precision which is # Relevant / # docs Retrieved. For a reference, you are required to compare against the baseline so that you have some reference of the performance improvement your methods could bring beyond the baseline. Please be fair in judging relevance by actually reading the contents of the reviews or at least reading the sentence where the aspect appears to see whether the opinion and the review are relevant to the query.

Note: For scoring an equivalent to A in this project, you are required to implement at least 2 additional methods beyond a baseline and one of them should have some novelty beyond the ideas explained in this draft.

6. Project Report and Code Submission

Finally, you are required to write a small report (6-7 pages) providing the details of your implementation of various ideas or any new idea you found to work better overcoming the shortcoming of the above ideas. You are also requested to submit your code along with your report.