Project 2: Review

Information Retrieval and Interation

Datalogisk institut, Copenhagen University (DIKU)

Oleksandr Shturmov
oleks@diku.dk

December 27, 2013.

The below is a review of an anonymous student submission of Project 2.

The names of the files reviewed are `GR3-PR2.pdf` and `GR3-PR2-code.py`. Their `md5sums` are `18d28963757253a8e041e70081bd3b39` and `36b2edff8e7d408668a41edfcad07045`, respectively.

- It's a good idea to state the Python version you developed against.
- The switch from exercise 1 to 2 is not very clear. Are you missing a heading?
- Consider instead of having the user add a root tag, add it via your Python script. This will make your solution more portable (clearly, TREC documents are not valid XML documents, so your solution should not assume that they are).
- You say — "As such a matching search query is not favored, if it matches the headline, as would normally be the goal when trying to evaluate search hits." — I'm not sure what you mean here. Could you perhaps also provide a source for your claim? My understanding is that a headline may not be sufficient to evaluate the relevance of a document to a query.
- Your code doesn't split words as you state in the report. You state "splitting each document on known separators, such as space, newline, commas etc." Your Python script, on the other hand, picks out those sequences of characters that match the regular expression `[\w']+`, which without `LOCALE` and `UNICODE` flags is the same as `[a-zA-Z0-9_']+`. This means that words like "follow-up" are regarded as two words "follow" and "up", and numbers like "0.5" are regarded as two numbers, "0" and "5". These discrepancies aside, it's perhaps worth a note that the dataset does not contain non-English characters.
- It is not a good idea to check whether a word is a stopword after stemming it. A stopword stemmed is not necessarily the same word. My understanding is that you should either stem the stopwords, or check whether a word is a stopword before stemming it.
- You can shorten `d[x] = d[x]+1` to `d[x] += 1`.
- It can be confusing to the reader when you write something like $number\ of\ docs(10)$ in a formula, as this then looks like a function. Consider instead $number\ of\ docs = 10$, or just state this in the text somewhere instead.
- It is a little bit misleading to the reader of the code to temporarily call the document frequency `inverse_document_frequency`. This variable is soon overwritten with the actual inverse document frequency, but until then should perhaps be called something else.
- Consider defining functions rather than duplicating code.
- Consider reading in the queries rather than hard coding them in your Python script, again, to make your solution more portable.
- Instead of comparing log probabilities (you discuss e.g. why it's okay that they are negative), you could take the values and raise e to their power to obtain true probabilities (values between 0 and 1). For instance, $e^{-20.014} \approx 2.03 \cdot 10^{-9}$, which is very close to 0. However, you don't really gain anything other than avoiding weird negative values.
- It would be nice to guide the reader to your results. Since there was a Python script, I was expecting for the results to be easily reproducible with the help of the computer, yet you seem to have done the computations by hand. You should state that.
- You state that doing MAP for other lambdas would be "a waste of time". Can you provide an argument for why it the case that for other lambdas MAP wouldn't provide further insights?
- Otherwise, rather thorough overall.