An overview of the Tag Recommendation Algorithm

TRA is a Python implementation of a query expansion model in the context of tag-based text search. Suppose we have a body of text documents. Each document is tagged with a list of words which are representative of the content of the document. Examples include hashtags in a tweet, key words in an article or academic paper, tags in a Stack Exchange post etc.

Suppose we wish to search for documents concerning a given topic by specifying a tag which corresponds to this topic. For example, we might be interested in tweets concerning the 2014 NCAA basketball tournament. A logical choice of hashtag to use for our search would be #ncaatournament. However, it is likely that there are many tweets whose subject is the NCAA tournament which are not tagged with #ncaatournament. These tweets probably contain tags related to #ncaatournament such as #marchmadness, #michigan, #louisville etc. TRA is an algorithm which will attempt to discover these related tags. With these related tags in hand we can expand our search terms to find more documents relevant to our original criteria.

TRA is a pure Python implementation that is powered by the Whoosh search and indexing library. TRA is primarily a word count based algorithm plus some basic NLP features. We will demonstrate how TRA works by continuing the example of the 2014 NCAA tournament.

STEP 0 – 50,000 tweets (taken on March 22, 2014) containing hash tags have been indexed using Whoosh. Stop words have been removed and stemming has been employed to improve search performance.

STEP 1 – Search the index for all tweets containing '#ncaatournament'. There are 67 results. Here are some example tweets:

- louisville did louisville things to win today's game #marchmadness #ncaatournament
- i knew picking #syracuse to win two games in #ncaatournament was asking for trouble
- moments away from tip let's go michigan #ncaatournament
- its raining 3's for michigan #ncaatournament

STEP 2 – From these results TRA extracts all tags that co-occur with #ncaatournament. These are highlighted in blue and will be considered candidates for recommendation.

STEP 3 – Considering the results from Step 1 as a single block of text, extract the 'key words' from this block of text using a query expansion model[1]. These are the words that appear relatively more frequently in the block of text as opposed to the body of all documents as a whole. These words are highlighted in red.

STEP 4 – Run a second search against the tweet index using the key words as the query terms. Calculate a similarity score[2] between the tweet and the query. Here are some example results:

- i'm just waiting for this texas michigan game to start #ncaamarchmadness2014 (score = 10.16)

[1] We use Whoosh's default query expansion algorithm. This is based on the Bose-Einstein statistics in the divergence from randomness framework. See here for more information.
[2] We use Whoosh's default BM-25F scoring algorithm. See here for more information.

- win or lose this is one of most exciting game i've ever watched #badgers #wisconsin (score = 7.98)
- michigan is on fire #goblue #marchmadness (score = 7.18)

STEP 5 – Extract all tags occurring in tweets which are similar to tweets containing #ncaatournament. Using a default score for the co-occurring tags, rank all appearances of co-occurring and similar hashtags for final recommendation.

In this use case the top two suggestions output by TRA are #marchmadness (1189 tweets) and #ncaamarchmadness2014 (130 tweets). Expanding the original search query of '#ncaatournament' to a query of '#ncaatournament OR #marchmadness OR #marchmadness2014' results in a 20x increase in the number of tweets returned.

TRA is fairly straight forward in concept, yet it exhibits good performance in a practical application to the Twitter dataset.