Building Bridges across Social Platforms: Answering Twitter Questions with Yahoo! Answers

Mossaab Bagdouri
Department of Computer Science
University of Maryland
College Park, MD 20742, USA
mossaab@umd.edu

Douglas W. Oard
iSchool and UMIACS
University of Maryland
College Park, MD 20742, USA
oard@umd.edu

ABSTRACT

This paper investigates techniques for answering microblog questions by searching in a large community question answering website. Some question transformations are considered, some proprieties of the answering platform are examined, how to select among the various available configurations in a learning-to-rank framework is studied.

CCS CONCEPTS

•**Information systems** →**Question answering;** *Test collections;*

KEYWORDS

Microblogs; CQA; Cross-platform question answering

1 INTRODUCTION

Over 81% of the questions asked on the microblogging service Twitter that are not addressed to a specific user receive no response [7]. For questions that express a true information need, any useful answer might be highly appreciated. Unanswered questions can be handled by suggesting answers to similar prior questions [9] or by routing the new question to some relevant expert who might be willing to provide an answer [5]. This approach has been extensively investigated using questions previously posted to the same platform where the new question has been posted. Well known techniques leverage features that can be extracted from old questions and answers, as well as the social graph between the users, the questions and the answers.

Sometimes, however, it might be better to look elsewhere for the answer. Community Question Answering (CQA) websites such as Quora and Yahoo! Answers have became very popular in the last decade, gathering hundreds of millions of questions with their answers. This makes them a suitable place to find answers for questions that have been posed elsewhere. In this paper we use a large crawl of Yahoo! Answers to search for threads that are potentially useful for a tweet question (Section 3), we compare the importance of different fields in which we can search (Section 2), and we study some approaches for adapting the language of Twitter questions to that of Yahoo! Answers (Section 2.1). We present our results in Section 4 before concluding with an overview of future directions that can benefit from our release of the annotations (Section 5). To the best of our knowledge, this is the first work to examine the usefulness of a CQA service for answering questions posted on a microblogging service.

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

978-1-4503-5022-8/17/08…$15.00
DOI: 10.1145/3077136.3080755

2 METHODS

In our search task, we want to retrieve a "thread" (i.e., an old question with its answers) from Yahoo! Answers that would be useful for answering the question newly posted to Twitter. A thread has several fields in which we can search. A reasonable baseline is the concatenation of the title and body of the question, together with all of its answers. This approximates a simple search for a web page in a search engine. Alternatively, we can index each field separately. This allows us to study the importance of each field independently from the others, and to examine different combination possibilities. We implement this alternative using BM25 [8].

There are two possibilities for indexing the fields of a thread. In the first, we index each field of the question, and the concatenation of all of its answers. We call this indexing setup Question-per-Document (QpD). In the second, the indexed document contains the two question fields and a single answer. That is, we index as many documents for a given thread as there are answers. We call this indexing setup Answer-per-Document (ApD). We refer to the indexed fields as question title (T), question body (B), title and body concatenation (C), and answer(s) (A).

We experiment with various combinations of these four indexed fields. The weight of each field is by default set to 1, but we also perform a two dimensional grid search on the weights of the B and A fields in the QpD-TBA configuration (i.e., Question-Per-Document, indexing the Title and Body separately along with a single Answer). For each configuration, we score the the top-1 thread, breaking ties (which is often needed when searching only the T field) by selecting the most recent thread.

2.1 Question Rewriting

Tweets have characteristics that are less common in some other platforms, some of which we address in this section.

2.1.1 Hashtag Segmentation. Twitter users often use hashtags to highlight some notion. Since Twitter hashtags don't contain spaces, it is common to concatenate the terms of a multi-words expression. Sometimes a *CamelCase* convention is used, as in "i wonder if #TheBible is or will be on Netflix?" In other instances,

no capitalization cues are present, as in "Is she Ilona or Elona? #thearchers." We expect hashtag segmentation to improve retrieval effectiveness. Our segmentation approach has three stages. In the first, we remove the # symbol, and use Google's cloud natural language API[1] to see if the resulting term is classified as an entity (although we ignore entities of type OTHER). This stage aims to avoid segmenting single-word proper names such as "Washington." In the second stage, we generate one or more candidate segmentations. If a hashtag follows the CamelCase convention (detected with a regular expression), we segment at capital letters. Otherwise, we use the vocabulary of our Twitter index (Section 3.3) to extract all possible segmentations (deleting any candidates containing 4 or more words).[2] Since some segmentations may be unreasonable (e.g., segmenting #iPhone into "i phone"), in the third stage, we remove segmentations that appear (in order) less frequently in our Twitter index than the hashtag (without the #). If no segmentation passes this filter, we maintain the hashtag (without the #). Otherwise, we replace it with the segmentation that has the highest frequency (breaking ties arbitrarily).

2.1.2 Spelling Correction. Twitter is mostly accessed from mobile devices[3] on which small keyboards increase the chance of misspellings. Consider for example "Why did the great awaking happen?" We have little hope for finding an answer unless the spelling of *awaking* is corrected to *awakening*. This problem is particularly critical when the misspelled word is a key term in the question. Another impact appears when a high frequency word (e.g., a stop word) is misspelled, typically resulting in a rare word with high IDF. For example, because we lowercase everything before performing a search, "should igo to school tomorrow?" leads to the undesirable retrieval of threads about Inter-Governmental Organizations (IGO).

We perform spelling correction in three steps. As with hashtag segmentation, we first exclude terms that are classified as entities (of a type other than OTHER). We then generate a list of (up to) the 1,000 closest words by Levenstein distance, using a model trained on character n-grams from our Twitter index.[4] Finally, we keep only alternatives for which both their document frequency and the document frequencies of the terms to their left and right (up to the first stopword) are greater than those of the original word. If any alternatives pass this filter, we return the alternative with the highest document frequency as the possible correction. To limit the effect of correction mistakes, we treat the possible correction as a synonym of the original word, computing the BM25 score for each field after summing the term frequencies of the original term and its possible correction and approximating the combined document frequency with the maximum of the two document frequencies (which is the document frequency of the possible correction).

2.1.3 Synonyms. The informal language of tweets encourages the adoption of some writing conventions that are less frequent in other platforms. For example, *you* and *conversations* would be synonyms to *u* and *convos* in "Should u read your kids convos on the Internet?" We find synonyms in three stages. The first and the third stages are identical to what we do for spelling correction. For the second stage (suggesting a candidate synonym), we use a word2vec [6] model trained on our Twitter corpus to suggest the nearest word to the original one, but only if the cosine similarity of their vectors exceeds an arbitrary threshold of 0.5.

[1] https://cloud.google.com/natural-language
[2] We use the WordBreakSpellChecker.suggestWordBreaks() method of Lucene 6.3.
[3] http://venturebeat.com/?p=2014007
[4] We use the SpellChecker.suggestSimilar() method of Lucene 6.3.

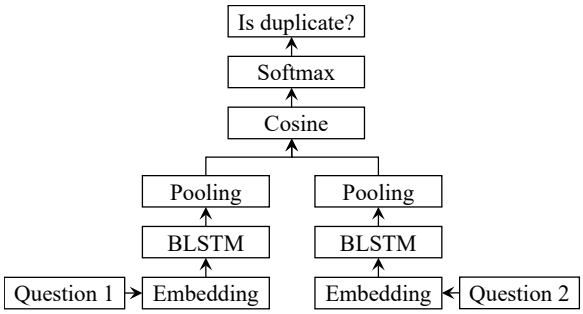

Figure 1: An architecture for detecting duplicate questions.

2.2 Term Statistics

The importance of a term is indicated, in the BM25 scoring function, by its IDF. As a result, the same term might have different IDF values in different corpora. For the question "What am I gunna do with this dog for the night?" we observe that *night* has a high IDF in Yahoo! Answers compared to *dog* The opposite (and we think more desirable) relative IDF rank is true for Twitter, however. Some words seem to suffer from a "cost of fame" in which they are so important that many questions are asked about them in Yahoo! Answers, (where there is an entire subcategory for dogs), thus diminishing their IDF. To mitigate this effect, we can use the IDF statistics from our Twitter index.

2.3 Question/Question Similarity

Similar questions might be phrased in different ways, so we need some way of measuring the extent to which a Twitter question is similar to a question in Yahoo! Answers. Quora has recently released a corpus of 404,351 pairs of questions, among which 149,306 are indicated to be duplicates.[5] We use 90% of those pairs to train the neural network depicted in Figure 1, and the remaining validation subset to stop training when the accuracy does not improve over the best prior results in the previous 10 epochs. We return the model that has the best accuracy (85.5%) on that validation set, after optimizing it with ADAM [4], using mean squared error as a loss function as implemented in Keras.[6]

2.4 Selecting Configurations and Answers

The approaches we have introduced so far aim to find configurations that work the best on average. However, it is possible to use the features of the questions and the answers to select the thread to be retrieved given the votes of different configurations, and our prior knowledge of their average performance. Here we present a three-stage process to select the best thread among those returned by several configurations.

[5] http://qim.ec.quoracdn.net/quora_duplicate_questions.tsv
[6] http://keras.io

2.4.1 Ordering the Configurations. Let N be the number of available configurations, $C_1, \ldots C_{n \leq N}$ be a subset of configurations, and $T_1, \ldots T_m$ be the number of training questions. Every pair (C_i, T_j) corresponds to a retrieved thread with a score $S_{i,j}$. The maximum average score, over the training questions, that can be achieved given this combination (with an oracle) would be: $\hat{S} = \frac{1}{m} \sum_{T_i} \max_{C_j} S_{i,j}$. Our goal is to find the subset of configurations that maximizes this value given n. With as many as 74 configurations in our experiments, a greedy search is considerably more efficient than exhaustively trying every possible combination. We start by finding the best single configuration. We then repeatedly use the best combination we obtained at iteration $n-1$, which gave us potential average score $\hat{S}_{n-1}$, and iterate over the remaining configurations to maximize $\hat{S}_n$. This process yields an ordered list of configurations that can be added, one at a time, to form several combinations.

2.4.2 Learning to Rank Threads. With some combination of configurations that, collectively, retrieves a set of threads, we want to learn to rank those threads. For every pair of a question and a retrieved thread, we extract the following vector of features:

- The BM25 scores of the title, the body, and the answer(s).
- The neural similarity scores between the question, and each of the three fields above.
- The number of answers in the thread (log scaled).
- The min, max, mean and standard deviation of the scores of each answer, both for BM25 and neural similarity.
- The number of threads with the same BM25 score as the candidate (log scaled)
- All the same features, using the rewritten question (with the three rewriting operations).
- Binary indicators of whether each configuration returned that thread.

Given a training question with several threads, we integerize the ground truth score for each thread (Section 3) using 0.5, 1.5 and 2.5 as cutoff points to produce scores of 0 (bad), 1(fair), 2 (good) or 3 (excellent). Finally, we train a learning-to-rank (L2R) model based on those threads using the SVM^{rank} software [3].

2.4.3 Selecting the Best Combination of Configurations. Given an ordered list of configurations (Section 2.4.1) and a model for ranking the threads of a particular combination of configurations (Section 2.4.2), we can select the best combination. To do so, we start with the best single configuration, and record its effectiveness on the training and validation questions, considering it to be the best combination so far. Then, we iterate over the ordered configurations, one at a time, adding each to the pool of configurations, and training its L2R model. We record the average score of the predictions on the training and validation questions using the actual ground truth scores (not the integerized versions). If the effectiveness increases in both sets, we consider the actual combination to be the best one. We stop when we finish our enumeration, and return the most recent best combination.

3 TEST COLLECTION

We present a set of Twitter questions, a crawl of Yahoo! Answers, and a collection of tweets used to build a language model.

3.1 Questions and Answers

Among questions with real information needs, only a small fraction could reasonably be answered by an automated system. Consider, for example, "@user hey, when u coming back?" Clearly, the asker would want an answer to this question, but probably only the mentioned user could provide it. It would be advisable for an answering system to skip such questions. We have collected a set of 5,000 questions posted on Twitter in February 2016 and asked annotators on the crowdsourcing platform CrowdFlower to indicate whether some stranger probably exists who could read the question and offer a useful answer. In this paper, we use the resulting set of 362 tweets deemed to be answerable questions, with a 177/85/100 split between training, validation and test.

With the large base of old questions and answers available in Yahoo! Answers, we hope to successfully find useful answers to a substantial number of questions asked on Twitter. One option would be to issue the question as a query and rely on the questions and answers retrieved by its"black box" internal search engine. However, this would prevent us from studying the different options for building and using the inverted index. Thus, we obtain a crawl of 123M questions and 673M answers from [2]. We exclude questions and answers that contain any term from a "dirty word" list,[7] and index all of the remaining questions and answers posted prior to 2016 (to avoid "leaking" future information to new answers).

3.2 Ground Truth

We also collect annotations for the results of our experiments using CrowdFlower. Because our task is similar, we adopt the same 4-level relevance scale (bad, fair, good, excellent) as the the TREC LiveQA track [1]. We assign the weighted average score over three annotators (where the weight is computed from annotator accuracy on a set of questions with known answers) as the ground-truth relevance score of the thread. Annotators with accuracy scores below 85% were removed and replaced.

As assessing all answers to a question might be impractical when many answers exist, we present only the question title, body, and a what we expect to be the best few answers. We select these answers in part based on metadata from Yahoo! Answers and in part based on whichever of our systems found the answer. We first select the best answer, as designated by the asker, if one exists. Otherwise, we select the answer with the highest difference between thumbs-up and thumbs-down votes, breaking ties by the score assigned by the system that found that answer. We also include whatever answer our system scored highest. In both cases, if multiple systems retrieve the same thread but disagree on which is the best answer, we include the best answers from each.

We obtained the annotations in several batches. In each batch we gathered the annotations for all threads that had not been previously assessed for all 362 questions. This allowed us to use the results of prior annotations to incrementally improve our systems, thus generating a richer test set, akin to the way systems from one year are used to guide the development of test collections in subsequent years at shared-task evaluations such as TREC. We note, however, that different annotators assessed different batches.

[7] Downloaded from https://gist.github.com/roxlu/1769577

3.3 Twitter Language Model

For language modeling, we obtained the Twitter random 1% public sample stream between January 2012 and December 2015 from the Internet Archive.[8] We keep only English tweets and index them with Lucene (without stemming or stopword removal), recording the positions of each indexed term in its tweet. We use this positional index as a language model to guide our question transformations (Section 2.1).

4 RESULTS

We experiment with a combinations of 74 configurations. Table 1 shows the average top-1 accuracy (on a [0-3] scale) for some of the combinations. First, we observe that the single best field is the Title (line 1). It is significantly ($p < 0.05$, two-sided paired t-test) better than the Body and the Answer fields (lines 2 and 7) in all sets (i.e., training, validation and test). Searching using this field is also better than searching using the entire page as a single field (line 8), with a significance observed in the training and test sets. It also appears that question-per-document indexing may be a bit better than answer-per-document indexing (compare lines 5, 6 and 7 to 9, 10 and 11), but weak significance is observed only in the training set ($p < 0.1$). Tuning the weights of the fields (line 12) seems to overfit to the training set, where it is significantly ($p < 0.05$) better than all combinations other than QpD-T (i.e., lines 2 through 11). On the validation and test sets, this tuning is not better than some of those combinations. None of the query rewriting methods, individually or in combination, improve the results significantly, and the same is true for using the IDF of the Twitter index. Insignificant positive differences, when observed, are restricted to the training set. The statistically significant improvements we observe ($p < 0.01$) with the L2R model over all configurations appears to be an instance of overfitting. In fact, the results over the validation set decrease slightly (from 1.48 of line 5 to 1.45 in line 26), and the gain we get in the test set (from 1.32 to 1.37) is not statistically significant.

Table 1: Effectiveness of configurations over the scale [0-3].

#	Configuration	Fields	Average score		
			Train.	Valid.	Test
1	BM25	QpD-T	1.22	1.19	1.32
2	BM25	QpD-B	0.80	0.89	0.75
3	BM25	QpD-TB	1.10	1.14	1.05
4	BM25	QpD-C	1.13	1.16	1.20
5	BM25	QpD-TA	1.20	**1.48**	1.21
6	BM25	QpD-TBA	1.11	1.22	1.14
7	BM25	QpD-A	0.79	0.82	0.74
8	BM25	QpD-P	0.85	1.09	0.88
9	BM25	ApD-TA	1.10	1.27	1.12
10	BM25	ApD-TBA	1.02	1.15	1.04
11	BM25	ApD-A	0.54	0.68	0.41
12	Weighted BM25	QpD-TBA	1.32	1.28	1.32
13	BM25 + Hashtag Split	QpD-T	1.24	1.19	1.31
14	BM25 + Hashtag Split	QpD-TA	1.22	1.46	1.20
15	BM25 + Spell Correction	QpD-T	1.23	1.19	1.30
16	BM25 + Spell Correction	QpD-TB	1.13	1.17	1.03
17	BM25 + Spell Correction	QpD-TA	1.21	**1.48**	1.19
18	BM25 + Synonyms	QpD-T	1.22	1.24	1.29
19	BM25 + Synonyms	QpD-TA	1.21	1.43	1.22
20	BM25 + 3 Rewriters	QpD-T	1.25	1.24	1.27
21	BM25 + 3 Rewriters	QpD-TA	1.23	1.41	1.19
22	BM25 + 3 Rewriters	QpD-C	1.12	1.17	1.10
23	BM25 + Twitter IDF	QpD-T	1.21	1.08	1.32
24	BM25 + Twitter IDF	QpD-TA	1.09	1.38	0.96
25	BM25 + Twitter IDF	QpD-P	0.82	1.15	0.97
26	L2R = (12) + (20) + (25) + (22)		**1.43**	1.45	**1.37**
27	Oracle		1.90	2.03	1.86

5 CONCLUSION

We studied the possibility of answering the questions asked on Twitter using Yahoo! Answers, and found that, on average, two thirds of the answerable questions do have an excellent answer there. We found that searching in the title field of the old questions yields a significant improvement over search in the concatenation of all the fields of a CQA thread. Small improvements are sometimes observed using various techniques, such as the tuning the weights of the indexed fields, rewriting the tweet question, and using the IDF of an index of tweets. While none of these techniques is particularly better than the others, the pool of diverse threads they retrieve suggests that a failure analysis might help to identify techniques that can be employed for specific question types. We have released our test collection to encourage further investigation.[9]

6 ACKNOWLEDGMENT

This work was made possible in part by NPRP grant# NPRP 6-1377-1-257 from the Qatar National Research Fund (a member of Qatar Foundation) and by an IBM Ph.D. Fellowship. The statements made herein are solely the responsibility of the authors.

[8] http://archive.org/details/twitterstream

[9] http://cs.umd.edu/~mossaab/files/aqweet-answering.tgz

REFERENCES

[1] Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna Harman. 2015. Overview of the TREC 2015 LiveQA Track. In *TREC*. Gaithersburg, MD.

[2] Mossaab Bagdouri and Douglas W. Oard. 2015. CLIP at TREC 2015: Microblog and LiveQA. In *TREC*. Gaithersburg, MD, USA.

[3] Thorsten Joachims. 2006. Training Linear SVMs in Linear Time. In *KDD'06*. Philadelphia, PA, USA, 217–226.

[4] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR'15*. San Diego, CA, USA.

[5] Baichuan Li and Irwin King. 2010. Routing Questions to Appropriate Answerers in Community Question Answering Services. In *CIKM'10*. Toronto, ON, Canada, 1585–1588.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Workshop at ICLR'13*.

[7] Sharoda A. Paul, Lichan Hong, and Ed H. Chi. 2011. Is Twitter a Good Place for Asking Questions? A Characterization Study. In *ICWSM'11*.

[8] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *FnTIR* 3, 4 (April 2009), 333–389.

[9] Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the Past: Answering New Questions with Past Answers. In *WWW'12*. Lyon, France, 759–768.