CLIP at TREC 2016: LiveQA and RTS

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

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

ABSTRACT

The Computational Linguistics and Information Processing lab at the University of Maryland participated in two TREC tracks this year. The LiveQA and the Real-Time Summarization tasks both involve information processing in real time. We submitted eight runs in the total. In both tasks, our best system had the highest precision among all automatic participating systems. This paper describes the architecture and configuration of the systems behind those runs.

1. INTRODUCTION

We participated in the LiveQA and the Real-Time Summarization TREC 2016 tracks [1, 4] with systems that were derived from our earlier participation in the TREC 2015 LiveQA and Microblog tracks. Lessons learned from our previous participation were taken in consideration. We describe the systems for these tasks in the following two sections.

2. LIVEQA

We developed two systems for the LiveQA track, depicted in Figure 1. **CLIP-YA** searches in an extensive crawl of Yahoo! Answers (Y!A). **CLIP-TW** searches in a large corpus of tweets.

2.1 Answering with Old Yahoo! Answers

Last year, we obtained 226M questions and 1.3B answers through an extensive crawl of Yahoo! Answers. We searched in a subset of those to return answers to new questions. We updated this collection by recrawling all of the questions (to gather new answers and user identifiers) and revisiting the pages of all the users (to gather more questions and users identifiers). Then, we crawled all of the new questions and users through several iterations as explained in [2], yielding a total of 260M questions and 1.4B answers. We, then, limited our focus to the questions and answers downloaded from the main Yahoo! Answers website.[1] We used this subset of 123M questions and 673M answers for both searching for candidate answers (Section 2.1.1) and training a deep neural network that ranks these candidate answers (Section 2.1.2).

[1] `https://answers.yahoo.com`

TREC'16, November 15–18, 2016, Gaithersburg, Maryland USA.

The architecture of our system **CLIP-YA** is designed as a cascade of four stages. First, we search in the corpus to extract a list of candidate answers. Second, we score the answers based on a classifier trained on old question-answer pairs from the the Y!A crawl. Third, we rescore the answers based in part on this first score, using a classifier trained on TREC 2015 LiveQA qrels. Fourth, we optionally combine several short answers into a long one.

2.1.1 Searching in Old Yahoo! Answers

Preliminary evidence from our participation in the first edition of this track was in favor of using more terms from the question for searching for candidate answers. In fact, the average score of 0.82 for the 11 questions for which we had used both the title and the body of the question was higher than the average score of 0.62 for the 1,068 questions for which only the title of the question was issued to the search engine. However, long queries come with a high efficiency risk, as it might take over the allowed 60 seconds to extract the posting lists and the corresponding documents from disk. For this reason, we put our index on a solid-state drive (SSD), instead of a traditional hard disk drive (HDD). This allowed us to use all of the terms of the title and the body of the question without worrying about timing out.

From last year, as well, we found that the 186 cases in which we searched in the combination of the title and the body of the old questions had an average score better than the 109 cases in which we searched only in the title field (average scores of 0.54 and 0.43, respectively). Searching in old answers appears to be better than either approach, however, as the average score of 0.67 for the corresponding 784 cases indicates. Overall, it appears that the more content we search in, the better the result we can expect. Consequently, we decided to search in the combination of the title, body, and answer of old answers, for each incoming question.

The query and indexed documents are preprocessed by substituting spaces for all characters other than alphanumeric and apostrophe, lower casing, and applying the Porter stemmer [6] from Lucene 6.[2] A list of 100 candidate answers is returned, ranked by the BM25 implementation of Lucene.

2.1.2 Initial Scoring with Old Yahoo! Answers

The Y!A crawl is useful as a repository for searching for candidate answers. A retrieval model might be able to find topically relevant answers, but it might fail to find the good answers among those. Fortunately, there is content in Y!A that we can use to train a classifier to rank topically rel-

[2] `http://lucene.apache.org`

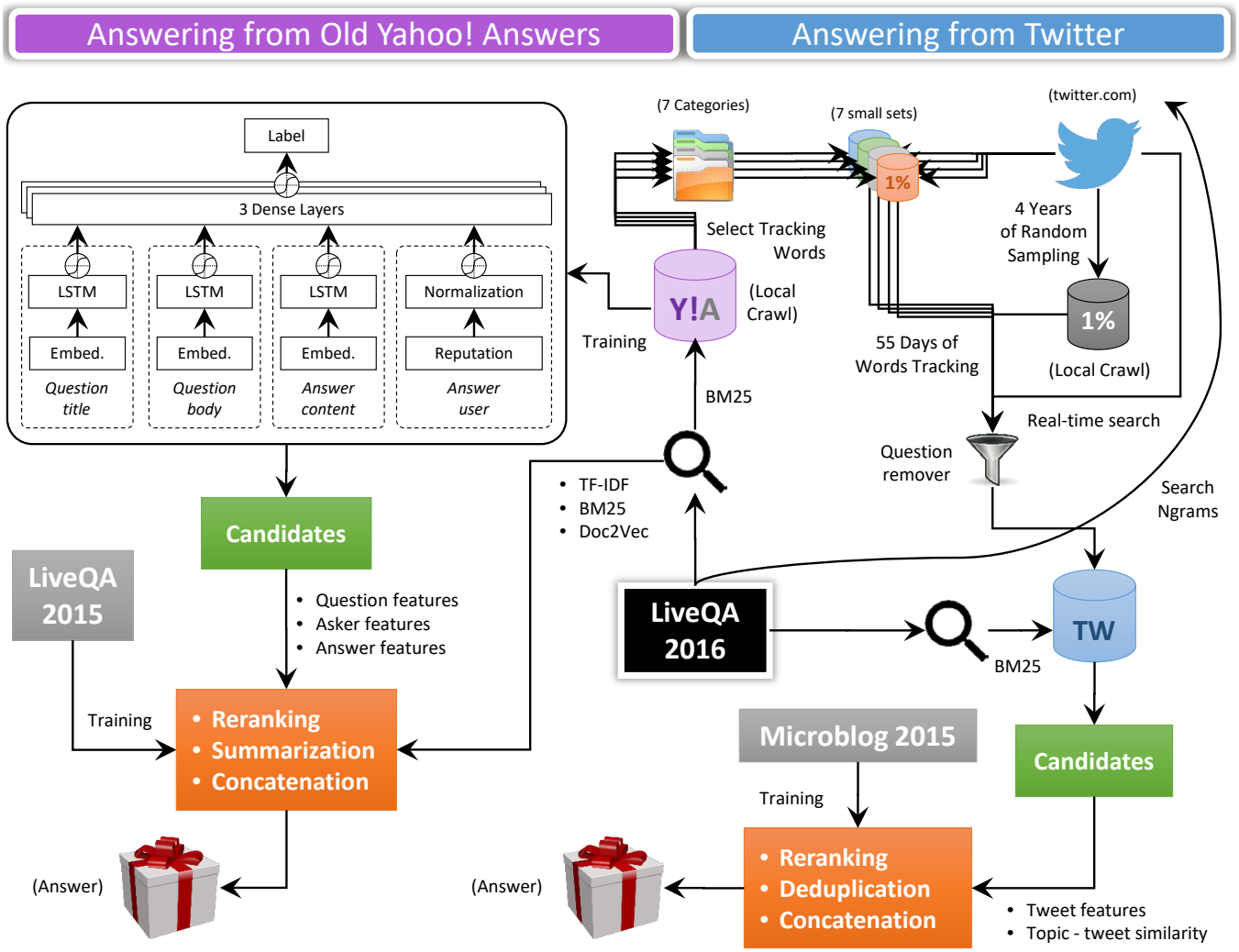

Figure 1: General architectures of the two LiveQA systems.

evant answers. For a given old question, we assume that all (or most) of its answers are relevant, but that some are more useful than others. We extract this usefulness from the social interaction of the crowd with the answers. The Y!A community can choose up to one best-answer for any given question. Y!A users can also vote for different answers by providing thumb-ups and thumb-downs. We define a ranking of the answers for any given question by ranking the best-answer, if one is available, at the top of the list. Then we sort the remaining answers in decreasing order according to the difference between the number of thumb-ups and thumb-downs, breaking ties arbitrarily.

The question then arises how best to select the training data, of negative and positive instances, on which we can train a classifier. Obviously, the answer at the top of the ranked list can be a positive instance. How best to select negative instances is, perhaps, less obvious. An answer ranked near the top of the list might actually be as good as the top one (consider a case where two identical good answers are present, but the website forces the asker to select no more than one best-answer). Some of the answers at the bottom of the list might be completely irrelevant to the question (e.g., spam). Hence, we decided to choose, as a negative instance, the answer located at the middle of the ranked list, after limiting ourselves to questions that have at least three answers.

Answers are often accompanied by user information. When this is the case, we extract the following seven integer features, which might serve as a surrogate of the reputation or the expertise of that user: the user level; the number of points; the number of questions, answers and best answers; and the numbers of friends and followers. Otherwise, we simply stuff that feature vector with zeros. Finally, each training instance is composed of the binary label for inferred utility, the title and body of the question, the answer content, and the seven-element feature vector for the answerer.

The top left corner of Figure 1 shows a deep neural network for training on this collection, which we implement in Keras.[3] Each text field is represented with an embedding layer of 200 dimensions, followed by an LSTM layer of 100 dimensions (the choice of LSTM was inspired by last year's best performing system [7]). Each of the user features is normalized to a value between 0 and 1, where the scaling

[3] `https://keras.io`

parameters are inferred from training. The three text layers and the user layer are, then, concatenated, forming a layer of 307 dimensions, which we connect to a stack of three fully connected layers of dimensions 100, 50 and 100, respectively, followed by the output layer (i.e., label of the answer). The sigmoid activation is applied between every pair of layers, as well as within the LSTM layer.

At prediction time, this network returns, for the title and body of the new question, and the content and the user of candidate answers, a score that we use in the rescoring stage (Section 2.1.3).

2.1.3 Rescoring with LiveQA 2015 Qrels

The process in Section 2.1.2 is useful for scoring old answers with respect to a new question. However, it does not use the similarity between the old and new questions. By crawling the URLs of the answers that last year's participants returned from Y!A, we construct a training corpus that contains, for each instance, the new question, the old question, the answer returned, and the annotated label. For each instance, we extract:

- Old question features: number of follows and answers.
- Old asker features: asker level (divided by 7), the ratio of the best answers that the asker has to all his answers, and the logarithms of one plus the asker points, questions, answers, friends and followers.
- Old answer features: whether the answer is a best answer, the number of thumb-ups and thumb-downs, the rating of the answer (a value provided by the asker between 0 and 5), and the count of comments that answer received.
- Similarity between the old and new questions: TF-IDF, BM25 and embedding-based similarities between the title, the body, their concatenation and the answer for the old question from one side, and the title and the body and their concatenation of the new question from the other side (i.e., $3 \times 4 \times 3 = 36$ similarity values).

With the SVM^{rank} software [3], we train a learning-to-rank classifier using the features above, in addition to the score returned from the neural network.

2.1.4 Answer Generation

The TREC LiveQA guidelines limit the answer length to a maximum of 1,000 characters. We summarize each candidate answer exceeding 1,000 characters in the following way. We split it into sentences based on periods and retain the first and last sentence, and as many of the sentences with the highest Jaccard similarity to the title of the question as possible until the 1,000-character limit would be exceeded by adding an additional sentence.

For candidate answers that contain less than 1,000 characters we take a different approach. Last year's best performing system often combined multiple answers into a single one [7]. This has motivated us to create a synthetic answer in the following way. We start with the first summary and then concatenate the subsequent summaries in the ranked list that have at least 100 characters, and for which, the concatenation would not violate the 1,000-character limit. This synthetic answer is what we return as a final answer.

2.2 Answering from Twitter

Answers might also be present in a platform different from the one where the questions were posted. As an illustrative example, we build a system that attempts to answer LiveQA questions from Twitter. In this section, we describe our approach for collecting a large number of tweets that are likely to contain answers to the seven categories of the track, we describe a method for ranking the tweets using a model trained on a the qrels of a different task, and we explain how multiple tweets were combined to form a single answer.

2.2.1 Collection

We want to construct a repository of tweets that contains a substantial number of tweets that are likely to be useful for answering LiveQA questions. Both of our Twitter based systems performed poorly last year, but we learned several lessons that guided our redesign of the Twitter based system for this year's participation.

Similarly to last year [2], we continued to use two major sources of tweets, which are (1) the tweets published through Twitter's sample stream (i.e., 1% of all public tweets) for over four years,[4] and (2) a selected set of tweets collected by tracking the terms of the *core vocabulary* of each category. Hoping to cover questions about current events, we add, this year, (3) a third source of tweets. From the title and body of the question, we extract word bi- and tri-grams. Using Twitter's search API, we issue a series of queries using those n-grams (starting with tri-grams first), requesting up to 100 tweets for each query. We stop when the total time spent in communicating with Twitter reaches 40 seconds.

We had found that searching in tweets that are detected to be not questions (system **CLIP-TW-A** of our participation in TREC 2015 LiveQA) yields an average performance better than that we get when we search in tweets detected to be questions and return their replies (system **CLIP-TW-Q**). Hence, our system this year (**CLIP-TW**), is based entirely on tweets detected to be not questions.

We had also found that, for system **CLIP-TW-A**, the average score of the 24 tweets retrieved from the small corpus of selected tweets (collected over a period of 3 weeks), was substantially better than the average score of the 781 tweets collected from Twitter's sample stream (0.46 and 0.19, respectively). Thus, we decided to collect more selected tweets. We did so over a period of 55 days.

We observed last year that, although the performance of our Twitter based system was substantially lower than that of our Y!A based system, the average score of the answers of the *Travel* category was comparable between **CLIP-YA** (2015) and **CLIP-TW-A** (0.29 and 0.25, respectively). Moreover, this category appears to be the most difficult one for the former, and the easiest for the latter. We, therefore, decided to give more attention to this category, and gathered more selected tweets. We did so by collecting the tweets using the terms of the 27 sub-categories of travel defined by Yahoo! Answers, in addition to the main travel category.

2.2.2 Rescoring with TREC 2015 Microblog Models

Our reliance only on the BM25 score last year might have contributed to the low performance of our Twitter based systems. This year, we enhance our scoring with a learning-to-rank (L2R) model. With a limited number of good Twitter-

[4]From `http://archive.org/details/twitterstream`.

Table 1: Performance of participating systems in the LiveQA task. This year, the average of all runs include manual runs.

Year	System	score (0-3)	succ@2+	succ@3+	succ@4+	prec@2+	prec@3+	prec@4+
2016	CLIP-YA	0.850	0.400	0.298	0.153	0.632	0.470	0.241
2016	CLIP-YA*	1.344	0.632	0.470	0.241	0.632	0.470	0.241
2016	CLIP-TW	-	-	-	-	-	-	-
2016	Average of all runs	0.643	0.329	0.212	0.104	0.422	0.271	0.131
2015	CLIP-YA (2015)	0.615	0.326	0.204	0.086	0.328	0.206	0.086
2015	CLIP-TW-Q	0.081	0.065	0.019	0.007	0.067	0.020	0.008
2015	CLIP-TW-A	0.144	0.102	0.034	0.008	0.138	0.046	0.011
2015	CMUOAQA	1.081	0.532	0.359	0.190	0.543	0.367	0.195
2015	Average of all runs	0.465	0.262	0.146	0.060	0.284	0.159	0.065

based answers, the TREC 2015 LiveQA qrels might be insufficient to train a useful L2R model. For this reason, we use a surrogate training corpus: the TREC 2015 Microblog Track. The topics of that track contain three fields: a short query (usually two to three terms), a description (in the order of a sentence), and a narrative (in the order of a short paragraph). For every <topic, tweet> pair, we extract the following features:

- Tweet features: word and stem counts and their ratio; the number of characters in the stemmed tweet; the presence of URLs, hashtags and mentions; and the logarithm of the ratio of number of followers to the number of friends.
- Topic - tweet similarity features: similarity value between the stemmed tweet on one side, and the stemmed topic description and narrative on the other side, using TF-IDF, BM25, Jaccard similarity and doc2vec similarity (where the document vector is the mean of the vectors of its terms, trained with word2vec [5]).

We apply the trained model to each LiveQA question by substituting the topic description with the question title, and the topic narrative with the question body. This produces a ranked list of the candidate tweets.

2.2.3 Answer Generation

One of the possible explanations for the poor performance of our Twitter based system last year is the length of the tweet. In fact, we might have been penalized by restricting ourselves to a maximum of 140 characters, while 1,000 characters could have been used. For this reason, we consider returning a concatenation of several tweets, instead of only one. First, we remove near-duplicate tweets by running a single-link clustering algorithm using Jaccard similarity with a threshold of 0.6 (which was the best threshold we had obtained in our TREC 2015 Microblog participation). With the remaining ranked tweets, we create a synthetic answer, starting with the first tweet, and then concatenating the subsequent tweets that have at least six words, without exceeding the 1,000 characters limit.

2.3 Results

Similarly to last year's scoring measure, each answer was given a score of -2 by NIST annotators if the answer is unreadable. Otherwise, the annotators assigned a score between 1 (bad) and 4 (excellent). The following performance measures are reported:

- **score (0-3)** is the average score over all questions after transferring 1-4 level scores to 0-3, and giving unreadable answers a score of 0.
- **succ@i+** is the number of questions with a score of at least i, divided by the total number of answered and unanswered questions.
- **prec@i+** is the number of questions with a score of at least i, divided by the number of answered questions.

Unfortunately, we had a couple of bad surprises regarding the annotation of our answers. Although our internal logs show that **CLIP-YA** answered all of the 1,015 questions, each in less than one minute, the TREC 2016 LiveQA organizers reported having received only 642 answers. The case with **CLIP-TW** is even worse. We timed out in only 5 out of 1,015 questions, but all of the answers were missed from the annotation pipeline. We have no clue about the cause of the missed **CLIP-YA** answers. However, for **CLIP-TW**, we found that the Java default XML parser has a known bug prohibiting it from loading XML files with emojis.[5] Since emojis are present in the answers we have returned, we speculate that they raised an error that removed all of the Twitter-based answers from the annotation pipeline.

Table 1 shows the official scores of our **CLIP-YA** system, as well as the scores that we would have expected to see if all of our answers had been annotated, and if the unannotated questions had been of comparable difficulty to those that were annotated[6] (**CLIP-YA***). For reference, we show the average scores over all systems that have participated this year. We also show the scores of our three systems from our previous participation, the score of the best performing system over all teams (CMUOAQA) that year, as well as the corresponding average scores over all participating systems.

Looking at the score and prec@2+ columns, we see that if we were to assume that the questions are of comparable difficulty across years, our **CLIP-YA** system has improved substantially when compared to **CLIP-YA** (2015). Comparing with our old selves, the evaluation results doubled (compare 0.615 vs. 1.344 and 0.632 vs. 0.328). Compared to the best performing system of last year, our score is higher by 24.3% (compare 1.081 vs. 1.344), and our prec@2+ is

[5] `http://stackoverflow.com/questions/31867818`

[6] We are not aware of any systematic difference in question difficulty between the annotated and the unannotated set. The expected score is computed by multiplying the official score by the ratio of the total number of answers to the number of answers that were annotated: $1.58 = 1015/642$.

Table 2: Improvement of CLIP-YA between 2015 and 2016.

Category	2015	2016	Improvement
Beauty & Style	0.50	1.60	+220%
Pets	0.65	1.55	+138%
Health	0.77	1.50	+114%
Arts & Humanities	0.64	1.37	+114%
Sports	0.51	0.98	+ 92%
Home & Garden	0.60	0.96	+ 60%
Travel	0.29	0.63	+117%

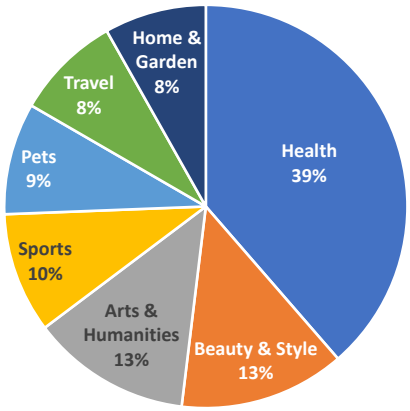

Figure 2: Distribution of questions over seven categories.

higher by 16.4% (compare 0.543 vs. 0.632). Finally, we observe that about two thirds of our answers (0.632) are at least fair, every second answer (0.470) is at least good, and one out of four answers (0.241) is excellent.

The improvement in the evaluation results of our **CLIP-YA** system between last year and this year is consistent across all categories, as can be seen in Table 2. We tripled our score for the *Beauty & Style* category, making it our best one (while last year it was second to the last). We doubled our score in *Pets*, *Health*, *Art & Humanities*, and *Travel*. But the latter is, still, the most difficult category suffering, perhaps, from a low prevalence in training (Figure 2).

3. REAL-TIME SUMMARIZATION

We participated in the push notifications task (a.k.a Scenario A), and the email digest task (a.k.a Scenario B) with automated systems similar to those we had developed last year for the Microblog Real-Time Filtering Track. The main differences, this year, are (1) we trained our reranker with the TREC 2015 Microblog qrels instead of the 2014 qrels, (2) we used the narrative field for the first time, and (3) we tried a new method for deciding whether a candidate tweet should be returned.

3.1 Tweet Scoring

A topic is represented as a triple of a title that contains a few keywords, a description that summarizes the topic in one sentence, and a narrative that consists of a paragraph that gives more details. We stem the topic fields with the Porter stemmer as implemented in Lucene 6.0 using its default list of stopwords. We use regular expressions to normalize all the tweets before stemming by removing emoticons, user mentions, URLs, retweet (RT) indicators, and punctuation.

Table 3: Performance of participating systems in Scenario B of the RTS task.

System	nDCG1	nDCG0
CLIP-B-0.6-2015	0.0718	0.0718
CLIP-B-0.6-MAX	0.1244	0.0173
CLIP-B-0.3-MIN	0.0312	0.0312

Similarly to last year, we expand the title field of the profile with the most similar terms using word-embeddings trained on a corpus of tweets downloaded from Twitter's sample stream and a probabilistic structured query scoring scheme based on Okapi BM25. Terms from longer fields (i.e., description and narrative) were not expanded. Instead, we used BM25, Jaccard similarity and a cosine similarity based on a doc2vec representation (where the document vector is the mean of the embedding vectors of its terms). Those query-tweet similarity scores, as well as tweet-specific features (count of stems, count of stems that are not stopwords, ratio of the previous two features, count of characters in the stemmed tweet, count of URLs, count of hashtags, and count of user mentions), and a sender feature (log of the ratio of the number of followers to the number of friends) are used to train a learning-to-rank classifier with SVM^{rank} software [3] using TREC 2015 Microblog qrels.

3.2 Near-Duplicate Detection

Our near-duplicate detection algorithm is identical to last year's. We limit ourselves to tweets that have a score above a manually selected score threshold (described below). Then, we apply single-link clustering with Jaccard similarity and some manually selected clustering threshold (described below) to group near-duplicate tweets. A new cluster is created whenever a new tweet has a similarity value below the clustering threshold with all of the previously seen tweets.

3.3 Deciding when to Answer

Deciding when to set the cutoff point for returning candidate tweets is a difficult task. We tried to estimate that cutoff point in the following way. Given the title of a profile, we issue all of its terms as a query to Twitter.[7] If no tweet is returned, we consider the union of tweets returned from issuing subqueries in which one term of the title was removed, and so on. We score all of the returned tweets as explained in Section 3.1. We compute the minimum, mean and maximum scores, which give us three possible relevance thresholds to be used to decide whether a tweet should be returned during the evaluation period.

3.4 Results

Table 3 shows the performance of our three systems that participated in Scenario B. **CLIP-B-0.6-2015** is the best system we had used in our TREC 2015 Microblog track, which had a clustering threshold of 0.6, and always returned 100 tweets per day. **CLIP-B-0.6-MAX** is our 2016 system with a manually selected clustering threshold of 0.6 that uses the maximum relevance threshold for deciding when to return a tweet. **CLIP-B-0.3-MIN** is our 2016 system with

[7]Instead of using Twitter's search API, which is limited to tweets posted in the last two weeks, we scrape the web page `https://twitter.com/i/search/timeline?q=[QUERY]`.

Table 4: Performance of participating systems in Scenario A of the RTS task.

System	relevant	redundant	not relevant	unjudged	length	P(strict)	P(lenient)
CLIP-A-0.7-MAX	91	1	89	507	679	0.5028	0.5083
CLIP-A-0.5-MEAN	158	7	171	911	1,227	0.4702	0.4911
CLIP-A-0.5-0	170	7	189	1,071	1,418	0.4645	0.4836

Table 5: Performance of participating systems in Scenario A of the RTS task.

System	EG1	EG0	nCG1	nCG0	GMP.33	GMP.5	GMP.66	mean latency	median latency
CLIP-A-0.7-MAX	0.2366	0.0206	0.2254	0.0093	-0.0950	-0.0629	-0.0328	227,092	178,997
CLIP-A-0.5-MEAN	0.2407	0.0354	0.2382	0.0328	-0.2556	-0.1656	-0.0809	121,940	12,090
CLIP-A-0.5-0	0.2397	0.0361	0.2415	0.0380	-0.3149	-0.2085	-0.1083	122,959	3,346

a manually selected clustering threshold of 0.3 that uses the minimum relevance threshold for deciding when to return a tweet. After the results were disseminated, we found a bug in the doc2vec similarity component that caused the scores to be very low.[8]

Tables 4 and 5 show the performance of our three systems that participated in Scenario A. **CLIP-A-0.7-MAX** has a manually selected clustering threshold of 0.7; it uses the maximum relevance threshold for deciding when to return a tweet. **CLIP-A-0.5-MEAN** and **CLIP-A-0.5-0** both have a manually selected clustering threshold of 0.5. The former uses the mean relevance threshold for deciding when to return a tweet, while the latter uses a fixed relevance threshold of 0, shared between all topics.

The scores of Table 4 are based on assessments made by mobile assessors (i.e., volunteer users who installed an application on their smartphone, and were actually receiving the notifications in real time). In this realistic setup, our three systems scored among the top four automatic systems, with system **CLIP-A-0.7-MAX** scoring the highest among all automatic systems. Thanks to its high clustering and relevance thresholds, it attained high precision.

Our three systems ranked lower (between 6^{th} and 9^{th}, among 34 automatic systems) based on NIST assessors (Table 5). But interestingly, the system that did best in the first evaluation setup performed the worst in the second evaluation framework. This, perhaps, suggests that factors other than topical relevance (e.g., time of the day, fatigue) also affected the mobile assessors.

4. CONCLUSION

We have presented the general architecture and the implementation details for the six runs we submitted for the Real-Time Summarization (RTS) task, and the two systems we built for the LiveQA task. The results suggest that a per-topic rescoring threshold and a high clustering similarity threshold can, each, improve the performance of our RTS systems. We are satisfied with the performance of our Scenario A systems, both in terms of recall and precision. We plan to fix the bug we have in our Scenario B systems and evaluate their performance using the released annotations.

The lessons we have learned from our participation in the LiveQA track of last year proved to be useful. In particular, search using most of the content both in the live question and in the indexed corpus, the deep neural network, and the combination of multiple answers, all participated in the high performance of our system based on old Yahoo! Answers. With respect to the answers that have not been annotated (i.e., one third of the answers returned by the system based on old Yahoo! Answers, and all of the answers returned by our Twitter-based system), we will use the similarity of their content with the annotated answers to estimate our performance. To avoid those silent exceptions, we recommend that the next edition of the LiveQA track returns an acknowledgment message for each incoming answer.

[8]Due to a multi-threading concurrency error, we were computing the average vector representation of a tweet using words different than the ones that constituted that tweet. This problem did not arise in Scenario A because the tweets were processed sequentially.

ACKNOWLEDGMENT

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

5. REFERENCES

[1] E. Agichtein, D. Carmel, D. Pelleg, Y. Pinter, and D. Harman. Overview of the TREC 2016 LiveQA track. In *TREC*, Gaithersburg, MD, USA, 2016.

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

[3] T. Joachims. Training linear SVMs in linear time. In *KDD '06*, pages 217–226, 2006.

[4] J. Lin, A. Roegiest, L. Tan, R. McCreadie, E. Voorhees, and F. Diaz. Overview of the TREC 2016 real-time summarization track. In *TREC*, Gaithersburg, MD, USA, 2016.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Workshop at ICLR*, 2013.

[6] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. 1997.

[7] D. Wang and E. Nyberg. CMU OAQA at TREC 2015 LiveQA: Discovering the right answer with clues. In *TREC*, 2015.