# CLIP at TREC 2015: Microblog and LiveQA

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 Microblog Real-Time Filtering and the LiveQA tasks both involve information processing in real time. We submitted nine runs in total, achieving relatively good results. This paper describes the architecture and configuration of the systems behind those runs.

## 1. INTRODUCTION

We participated in Microblog Real-Time Filtering and the LiveQA TREC 2015 tracks. Efficiency was taken in consideration, as the topics of both of these tasks are time-sensitive. We describe the systems for these tasks in the following two sections.

## 2. MICROBLOG REAL TIME FILTERING

The TREC Microblog track is concerned this year with the task of real-time filtering.[1] A user has an interest in some broad topic, and wants to stay up to date in that topic using a stream of microblog posts. The setup is operationalized in two scenarios. In the *push notifications* scenario (Scenario A), a system should notify the user with novel and relevant tweets within a short time after they are first posted. However, the user should not be bombarded with too many notifications. A limit of 10 notifications per day is therefore enforced. In the *email digest* scenario (Scenario B), a system filters the tweets of a particular day, few hours after the end of that day, to retrieve a ranked list of 100 potentially relevant and novel tweets.

The track ran for 10 days from July 20 to 29, 2016 (UTC), and was based on Twitter's sample stream (i.e., a random 1% sample of all public tweets). We submitted six automatic runs, three for each scenario.

[1]`https://github.com/lintool/twitter-tools/wiki/TREC-2015-Track-Guidelines`

### 2.1 Components

Most of the components we built are shared between the systems that we designed for Scenarios A and B, with some adaptation (Figure 1). We describe those components in this section.

#### 2.1.1 *Relevance Model*

A topic is represented as a triple of a *title* that contains few keywords, a *description* that summarizes the topic in one sentence, and a *narrative* that consists of a paragraph that gives more details. In all of our systems, we restricted ourselves to the title and description fields. Hence, we do not refer to the narrative field henceforth. We stem the topic fields with Porter stemmer as implemented in Lucene 5.3,[2] using its default list of stopwords. We use regular expressions to normalize all the tweets by removing emoticons, user mentions, URLs, RT indicators, and punctuation, before stemming them.

Our relevance model is based on Okapi BM25 term weights and title expansion using word embeddings and probabilistic structured queries. We use 1.6B English tweets from a corpus covering about 4 years of a Twitter sample stream, accessible from the Internet Archive[3] to train a word2vec model [4] and to estimate the document frequency (DF) of each term. The word2vec model is used to expand the title query stems with additional similar stems using the Euclidian distance over 200-dimensional vectors.

Let $t_i$ be a stemmed query term in the title, $t_{i,j}$ one of the top $J$ stemmed terms similar to, but different from, $t_i$, with a similarity value of $P_{i,j}$; and $d$ an incoming tweet. The score of the expanded title query is computed as:

$$Score(d, Q_{Title,exp}) = \sum_i BM25(TF(t_{i,exp}, d), DF(t_{i,exp})),$$

where the expanded term frequency is estimated as:

$$TF(t_{i,exp}, d) = TF(t_i, d) + \sum_j P_{i,j} TF(t_{i,j}, d),$$

and the expanded document frequency as:

$$DF(t_{i,exp}) = DF(t_i) + \sum_j P_{i,j} DF(t_{i,j}).$$

The score for the description field is simply the BM25 model. That is,

[2]`http://apache.lucene.org`
[3]`https://archive.org/details/twitterstream`

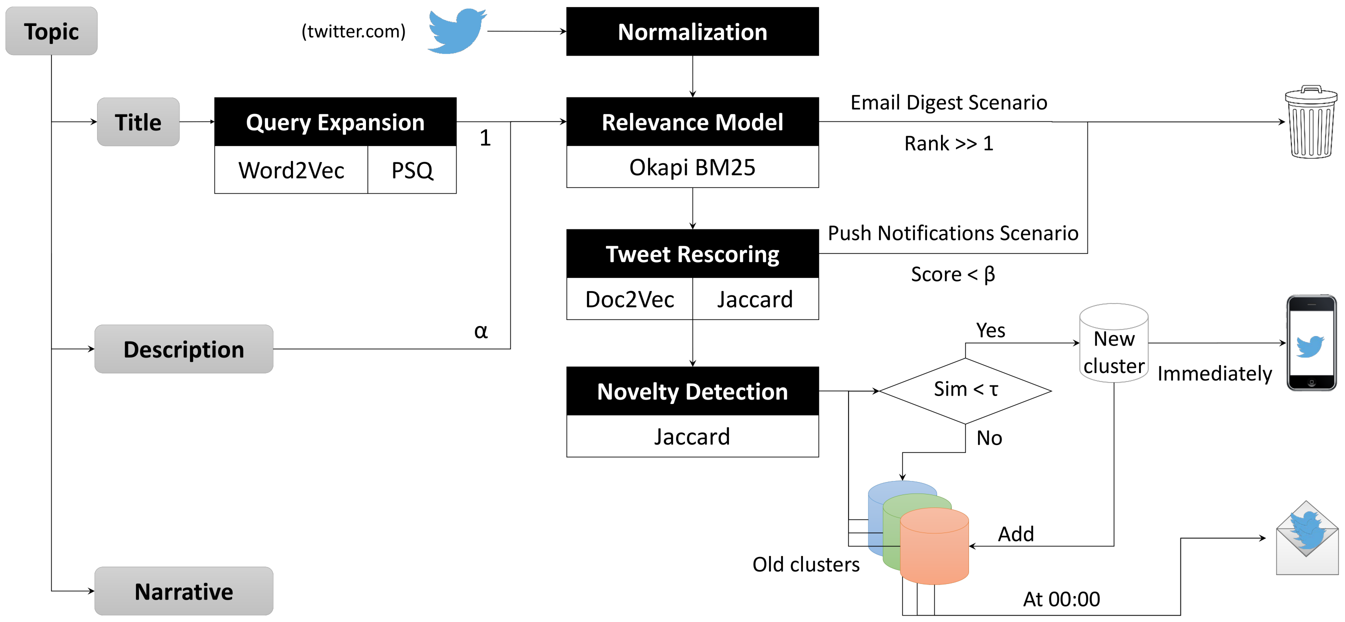


Figure 1: General architecture of the real-time filtering systems.

$$Score(d, Q_{Desc}) = BM25(TF(t_i, d), DF(t_i)).$$

Finally, the relevance score of a tweet $d$ for a topic $Q$ is defined with respect to its title and description as:

$$Score(d, Q) = Score(d, Q_{Title,exp}) + \alpha \times Score(d, Q_{Desc}).$$

We tune the parameters using a grid search on the TREC 2014 Microblog track topics. We set k1 = 0.09, b = 0.5 and average document length = 21 for BM25, $J = 5$ and $\alpha = 0.3$.

For Scenario B only, we eliminate tweets that are not ranked among the top 10,000 tweets at the end of this stage. The remaining tweets go to the rescoring stage, for both scenarios.

### 2.1.2 *Tweet Rescoring*

To refine the scores of the relevant tweets, we use the SVM$^{rank}$ package [2] to train a learning-to-rank model based on the TREC 2014 Microblog track topics, using the relevance score of the previous stage (Section 2.1.1), in addition to the following features:

- **Sender feature**: log of the ratio of the number of followers to the number of friends.
- **Tweet 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.
- **Tweet - query similarity features**: the two variants of the Jaccard similarity between the tweet and the description of the topic proposed by Magdy et al. [3], the BM25 similarity score between the expanded title and the tweet as explained in Section 2.1.1, and the cosine similarity between the doc2vec vector (i.e., the sum of the word2vec vectors) of the stemmed terms of the tweet and the doc2vec vector for the description field of the topic.

For Scenario A only, tweets that have a (re-scored) score less than a threshold $\beta$ are eliminated at the end of this stage. The remaining tweets go to the novelty detection phase, for both scenarios.

### 2.1.3 *Novelty Detection*

According to the task definition, a tweet is not considered interesting when the information it conveys has already been reported in an antecedent tweet that was present in the 1% public sample. We implement novelty detection with online single-link clustering based on the Jaccard similarity between the stemmed tweets. For each incoming tweet for topic $Q$ (other than those removed by scenario-specific processing following the relevance or rescoring stage), we assign the tweet to the cluster containing the most similar tweet, if the similarity exceeds certain threshold $\tau$. Otherwise, a new cluster is created and the incoming tweet is assigned to it. We maintain the same set of clusters for the entire 10 days of the live experiment since we don't want to return a relevant tweet if a similar one was returned even in a previous day.

In the case of Scenario A, the tweet is pushed to the user as soon as a new cluster is created. In the case of Scenario B, at the end of every (UTC) day, the tweet with the highest reranking score from each cluster from which no tweet has been reported on a prior day is selected and added to the ranked list for that day. In either scenario, that cluster is then marked so that it won't be used to suggest interesting tweets to the user (although it will keep gathering similar tweets, so that no new cluster with similar content is created).

## 2.2 Configurations of Submitted Runs

We submitted three runs for each scenario:

- **CLIP-A-5.0-0.5** and **CLIP-A-5.0-0.6** are two runs for Scenario A, where the score threshold (after rescor-

Table 1: Results for Scenario A of the Microblog Task.

| System | ELG | nCG |
|---|---|---|
| CLIP-A-DYN-0.5 | 0.1753 | 0.2426 |
| CLIP-A-5.0-0.6 | 0.1543 | 0.2221 |
| CLIP-A-5.0-0.5 | 0.1552 | 0.2193 |

Table 2: Results for Scenario B of the Microblog Task.

| System | nDCG |
|---|---|
| CLIP-B-0.4 | 0.2117 |
| CLIP-B-0.5 | 0.2420 |
| CLIP-B-0.6 | 0.2491 |

ing) was set for all topics to be $\beta = 5$. The clustering similarity score was set as $\tau = 0.5$ or $\tau = 0.6$ respectively.

- **CLIP-A-DYN-0.5** is a run for Scenario A, where the clustering similarity score was set to $\tau = 0.5$. However, the score threshold (after rescoring) was set per topic, and was tuned with a grid search, so that each topic had about 10 interesting tweets on July 18, 2015 (i.e., two days before the evaluation period).

- **CLIP-B-0.4**, **CLIP-B-0.5** and **CLIP-B-0.6** are three runs for Scenario B, where the clustering similarity score threshold was set to $\tau = 0.4$, $\tau = 0.5$ or $\tau = 0.6$ respectively.

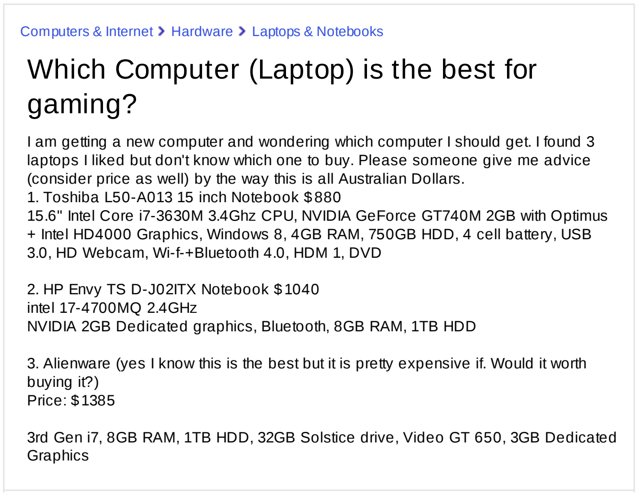
### 2.3 Results

The runs in Scenario A were evaluated using two metrics, each is based on gain and latency discount. The gain of a tweet $G(t)$ is a graded relevance score of 0 (not relevant), 0.5 (relevant) or 1 (highly relevant). A latency discount of $max(0, (100 - d)/100)$ is then applied to every tweet with a delay $d$ in minutes (rounded down).[4] The first metric is the expected latency-discounted gain (ELG), where the sum of the gains is divided by the number of tweets returned. The second metric is the normalized cumulative gain (nCG), where the same sum of gains is divided by the maximum possible gain.

Table 1 shows the results from our systems for this scenario. It appears that setting the relevance score threshold $\beta$ dynamically per topic was a good approach. Our best system (**CLIP-A-DYN-0.5**) ranked 21st out of 30 automatic runs based on ELG, but 3rd based on nCG. This suggests that our relevance model, perhaps, performed well. But we were penalized for attempting to return about 10 tweets a day per topic, while many fewer than that were typically actually relevant. It appears that a higher relevance score threshold could have led to a higher ELG score.

The runs of Scenario B were evaluated using nDCG@10. Table 2 shows the results from our systems for this scenario. It appears that setting a high clustering similarity threshold $\tau$ has a positive impact, on average. Our best system (**CLIP-B-0.6**) ranked 1st out of 36 automatic runs.

[4]Every tweet was processed in about one second. Thus, our gains were not impacted by the latency discount.


Computers & Internet > Hardware > Laptops & Notebooks

Which Computer (Laptop) is the best for gaming?

I am getting a new computer and wondering which computer I should get. I found 3 laptops I liked but don't know which one to buy. Please someone give me advice (consider price as well) by the way this is all Australian Dollars.
1. Toshiba L50-A013 15 inch Notebook $880
15.6" Intel Core i7-3630M 3.4Ghz CPU, NVIDIA GeForce GT740M 2GB with Optimus + Intel HD4000 Graphics, Windows 8, 4GB RAM, 750GB HDD, 4 cell battery, USB 3.0, HD Webcam, Wi-f-+Bluetooth 4.0, HDM 1, DVD

2. HP Envy TS D-J02ITX Notebook $1040
intel 17-4700MQ 2.4GHz
NVIDIA 2GB Dedicated graphics, Bluetooth, 8GB RAM, 1TB HDD

3. Alienware (yes I know this is the best but it is pretty expensive if. Would it worth buying it?)
Price: $1385

3rd Gen i7, 8GB RAM, 1TB HDD, 32GB Solstice drive, Video GT 650, 3GB Dedicated Graphics

Figure 2: Question asked under the category of Computers & Internet.

## 3. LIVEQA

The TREC LiveQA track[5] is a new task that loosely follows earlier TREC QA tracks, but with several substantial design changes. In this new track, the questions come in real time from real users, as posted on Yahoo! Answers.[6] This results in more natural and more diverse topics then was the case with earlier QA tracks in which the questions are developed by assessors or selected from query logs. LiveQA also incorporates an efficiency challenge, as the answers have to be provided in near-real time (specifically, in no more than one minute). A third challenge is that no document collection is provided, so participants must assemble any online or offline resources on which their systems will base their answers. These changes make the task more realistic and challenging.

Only one answer per question is judged for each participating system, using a relevance scale from 1 to 4.

This year, the LiveQA track focuses on eight categories, namely *Arts & Humanities*, *Beauty & Style*, *Computers & Internet*, *Health*, *Home & Garden*, *Pets*, *Sports*, and *Travel*. Figure 2 shows the example of a question[7] asked under the category *Computers & Internet*. We submitted three runs: one using old questions and answers from Yahoo! Answers (Section 3.1), and two using tweets (Section 3.2).

Figure 3 shows the general architecture of our LiveQA systems.

### 3.1 Answering with Old Yahoo! Answers

A study by Shtok et al. [5] has shown that about 25% of the titles of the questions posted on Yahoo! Answers in a 3-month period had occurred in a similar form (i.e., with a cosine similarity above 0.9) in a prior 11-month period. This suggests that it may often be possible to find similar questions that have previously been asked. Assuming that similar questions will have similar answers (which is not necessarily true, for instance, for generic or experience-based

[5]`https://sites.google.com/site/trecliveqa2015`
[6]`https://answers.yahoo.com`
[7]`https://answers.yahoo.com/question/index?qid=20130827015155AAwtYLQ`

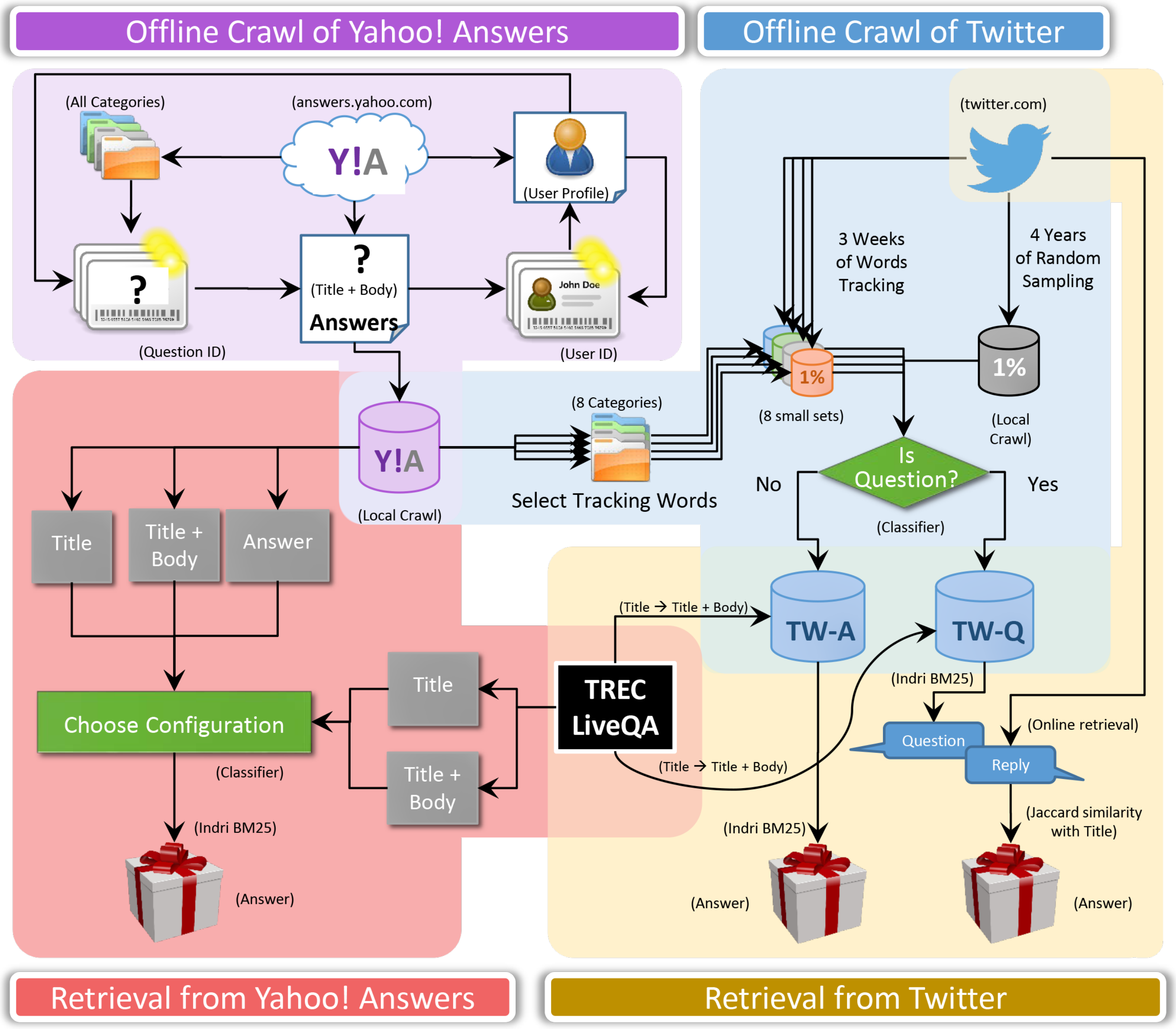


Figure 3: General architecture of the LiveQA systems.

questions), then a natural baseline system is one that simply does a search in old questions and answers for each incoming question. This section describes our process for crawling old questions and answers from Yahoo! Answers, and how we used the resulting collection to answer new questions (Figure 2).

### 3.1.1 *Crawling Old Questions and Answers*

To build a system that answers new incoming questions with the knowledge already existing in Yahoo! Answers, we decided to crawl all of the questions and answers that have ever been published on this platform, and that are still accessible (i.e., they did not get deleted). We do so in four steps:

1. Crawl all of the categories pages of the the main website, in addition to its 22 localized versions[8] to gather a fairly large set of question identifiers, and add them to a set $Q$. Each webpage shows up to 1,000 of the most recent questions.

2. Let $Q^*$ be the subset of questions in $Q$ that have not been crawled yet. Crawl the webpages of questions in $Q*$ to obtain the questions, their answers, and the user identifiers of those asking and answering the questions. Add the user identifiers to a set $U$.

3. Let $U^*$ be the subset of users in $U$ that have not been crawled yet. Crawl the webpages of users in $U*$ to acquire the identifiers of the questions they asked or responded to, which we add to the set $Q$, and to ac-

[8]e.g., `https://es.answers.yahoo.com`

quire the user identifiers for their friends and followers, which we add to the set $U$.

4. If either of $Q^*$ and $U^*$ is not empty, then go back to Step 2.

While this process has allowed us to gather a large set of 226M questions, 1.3B answers and 43M users, it is not guaranteed that we have obtained all of the data available in the website. This is the case for at least two reasons. First, the privacy settings of some users may be configured to hide the identifiers of the questions they asked or answered or the identifiers of their friends and followers. Second, some groups of users (especially the least active ones) might form an isolated clique that is not accessible by following links (based on questions or users) from the seed questions.

Once crawling is complete, we index the questions and the best answers found on the main Yahoo! Answers website, after stemming with the Porter stemmer. Crawling the other (localized) systems allows us to identify additional users, who may have also posted questions or answers to the main website, but we did not index the questions or answers from those localized websites because we do not expect many useful matches to be found there.

#### *3.1.2 Selecting Answers*

With a large corpus of prior questions and answers, we have several fields we can use for retrieval. Here we consider only the following six possibilities. For the incoming question, we use the title, but we also optionally concatenate it with the description. For the old questions, we consider searching in the title only, in the concatenation of the title and the description, or in only the best answer. When we search in old questions, we return its corresponding best answer. When we search in old best answers, we just return the best answer that we find. Because we did not have any ground truth for selecting among these alternatives in the first year of the track, we instantiated a small crowdsourcing task on CrowdFlower,[9] in which we showed the annotators questions from the final dry run, with up to six answers from the six retrieval configurations (when two or more methods returned the same answer, we would show fewer than six options). We allowed them to check-mark any answer they thought does indeed answer the question. Using annotations for 61 questions assessed by at least three trusted annotators for which at least one of them checked at least one of the answers, we trained a classifier to predict which configuration would be best for an incoming question. As features, we used the number of words and characters in the title and description fields in their stemmed and unstemmed versions, the category of the question, and the Jaccard similarity between the stemmed title and the stemmed description. We trained the cost-sensitive multiclass classifier of VowpalWabbit[10] using these features. Formally, for a training document assessed by $N$ annotators, let $v_{i,n}$ be the binary value implicitly indicated by annotator $n$ for one of the $I = 6$ retrieval configurations $i$. The cost $c_i$ associated with predicting the configuration $i$ is:

$$c_i = 1 - \sum_{n=1}^{N} v_{i,n} \Big/ \sum_{i'=1}^{I} \sum_{n=1}^{N} v_{i',n}.$$

[9]http://crowdflower.com

[10]https://github.com/JohnLangford/vowpal_wabbit

In other words, we assign a high cost for errors on questions for which all of the annotators agreed on the same answer, and a low cost for questions that have multiple good answers marked or high disagreement amongst the annotators. When a new question is received, we apply the trained model to choose which one of the six configurations to use to answer that question.

The guidelines limit answers to 1,000 characters. If the answer exceeds 1,000 characters, 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. This summarizes our submitted system **CLIP-YA**.

## 3.2 Answering with Tweets

Twitter, like several other popular social media websites, constitutes an enormous resource for information and opinions that are continuously produced by people around the globe. This makes it a potential place to find answers to questions asked on Yahoo! Answers. In this section we describe our approach to collecting tweets, and to using them as a basis for question answering.

#### *3.2.1 A Large Corpus of Random Tweets*

We consider Twitter as a resource for finding answers. As we do not have access to all of the tweets that have ever been posted, we need to find ways for increasing our chances of gathering tweets that will be useful to the expected questions. For this, we use two groups of collections. The first is a collection of tweets maintained by the Internet Archive Team since 2011 (Section 2.1.1). It consists of most of the JSON objects grabbed from a sample of about 1% of Twitter's public tweets. Because of some technical difficulties, tweets sent on some days are missing from this collection. Hence, we only have tweets from 1,215 days from the period September 27, 2011 to June 30, 2015. We added to this a collection of tweets that we streamed between July 27, 2015 and August 27, 2015. We denote this collection by $PublicStream$.

#### *3.2.2 A Small Corpus of Selected Tweets*

The Twitter API allows us to track[11] a set of up to 400 keywords.[12] When doing so, the API returns the tweets that contain at least one of these keywords, subject to the 1% limit computed over all tweets. For each of the eight categories, we think that selecting some keywords that represent its *core vocabulary* and then tracking Twitter content containing at least one of those keywords might give us a set of tweets that is richer (in terms of relevance to the potential questions) than the ones we would get by relying solely on the public stream.

We construct these eight core vocabularies following Fung et al. [1]. Formally, let a document in Yahoo! Answers be a question, its description or one of its answers. we denote by $DF(w_G)$ the document frequency of a word $w$ in the set of documents $G$. We then scale $DF(w_G)$ to a value between 0 and 1 as:

[11]https://dev.twitter.com/streaming/overview/request-parameters#track

[12]https://dev.twitter.com/streaming/reference/post/statuses/filter

Table 3: Words in the core vocabulary of each category.

| Arts & Humanities | Computers & Internet | Beauty & Style | Home & Garden |
|---|---|---|---|
| book | computer | hair | plant |
| books | windows | wear | paint |
| poem | laptop | color | wood |
| novel | download | skin | walls |
| writing | pc | makeup | plants |
| author | software | dress | garden |
| story | install | style | wall |
| twilight | click | | furniture |
| characters | files | | depot |
| authors | program | | soil |
| (13 more) | (11 more) | | (65 more) |

| Health | Sports | Pets | Travel |
|---|---|---|---|
| doctor | team | dog | travel |
| weight | football | dogs | city |
| diet | players | vet | trip |
| exercise | win | pet | hotel |
| body | teams | cat | airport |
| fat | fan | cats | flight |
| muscle | player | breed | hotels |
| pain | wwe | puppy | cities |
| eating | | pets | tourist |
| calories | | animals | places |
| (5 more) | | (13 more) | (14 more) |

$$df(w_G) = \frac{DF(w_G) - \min_{w' \in G} DF(w'_G)}{\max_{w' \in G} DF(w'_G) - \min_{w' \in G} DF(w'_G)}.$$

Finally, for a given category $i$, we denote by $H_i$ the value

$$H_i(w) = df(w_C) - df(w_{\overline{C}}),$$

where $C$ is the set of documents that belong to the category $i$, and $\overline{C}$ is the set of documents that do not belong to that category.

This value gives higher credit to words that are more frequent within the category $i$ than within all the other categories. In other words $H_i$ defines a ranking of words by their relevance to the questions and answers of the category $i$.

We observe that using the top 400 keywords for each category causes the Twitter API to send warnings for hitting the 1% maximum. Thus, we heuristically set the number of keywords, for every category, so that their filter matches about 1% of all the posted tweets. The final set of keywords for these eight categories is described in Table 3. We tracked these eight sets of keywords using eight Twitter accounts for three weeks (from August 7 to 27, 2015.) We denote each of the eight corresponding collections by $TrackedWords_i$, while $i$ corresponds to the name of the category.

Finally, when we receive a question of category $i$, we search in the union of *PublicStream* and $TrackedWords_i$, which we henceforth refer to as the search space $Corpus_i$.

### *3.2.3 Preprocessing*

We normalize all the tweets by removing emoticons, user mentions, URLs, RT indicators, and punctuation before stemming them with the Porter stemmer.

All of the questions have a title and most of them have a description as well. As both of these fields can be long, running the query as-is risks generating a high I/O load, and thus exceeding the limit of one minute per question. To mitigate this limitation we heuristically select the words of the query following these steps after stemming both the title and the description with Porter stemmer:

1. If the stemmed title has more than seven terms, we remove from them a list of 74 terms that we had manually selected from the most 100 frequent stemmed terms in the corpus of old questions and answers in Yahoo! Answers (Section 3.1).

2. We issue the preprocessed title as a query to an Indri[13] index corresponding to the category of the question, using the Okapi BM25 retrieval model.

3. We use the retrieved documents as a backup if the next stage does not complete within the allowed time limit.

4. We concatenate the processed title to the description field (processed in a similar manner), and issue the combined query to Indri.

### *3.2.4 Finding Answers*

By matching dry-run questions against the tweets in our corpora, we observed that several of the retrieved tweets are themselves actually questions. Such questions should not be returned as answers to the original question. This led us to make the distinction between two conceptual types of tweets: those that contain a question, and others (because those that do not contain a question might contain an answer). To implement this distinction, we extract two subsets from $Corpus_i$ for each category $i$:

- $Corpus_{i,q}$ is the subset of tweets that contain a question mark and are also detected to be questions seeking an answer (using a classifier that has an accuracy of 79% trained on normalized and stemmed 1-3-grams from tweets released by Zhao and Mei [6]). For $Corpus_{i,q}$, we start with the top 20 retrieved tweets. We then scrape their Twitter web pages to get all of their replies. We return the reply that is the most similar to the title of the incoming question, using the Jaccard similarity coefficient, and after processing them as described in Section 3.2.3. This is our submitted run **CLIP-TW-Q**.

- $Corpus_{i,a}$ is the subset of tweets that do not contain a question mark and that are also not detected (by that same classifier) to be questions seeking an answer. For $Corpus_{i,a}$, we simply return the tweet with the highest BM25 relevance score. This is our submitted run **CLIP-TW-A**.

## 3.3 Results

According to the guidelines, each system-submitted answer was given a score of -2 by the NIST annotator if the answer is unreadable. Otherwise, the annotator assigned it a score between 1 (bad) and 4 (excellent). The following performance measures are reported:

[13]http://www.lemurproject.org/indri.php

Table 4: Performance of participating systems in the LiveQA task.

| System | score (0-3) | succ@1+ | succ@2+ | succ@3+ | succ@4+ | prec@2+ | prec@3+ | prec@4+ |
|---|---|---|---|---|---|---|---|---|
| CLIP-YA | 0.615 | 0.993 | 0.326 | 0.204 | 0.086 | 0.328 | 0.206 | 0.086 |
| CLIP-TW-Q | 0.081 | 0.977 | 0.065 | 0.019 | 0.007 | 0.067 | 0.020 | 0.008 |
| CLIP-TW-A | 0.144 | 0.741 | 0.102 | 0.034 | 0.008 | 0.138 | 0.046 | 0.011 |
| All runs | 0.465 | 0.925 | 0.262 | 0.146 | 0.060 | 0.284 | 0.159 | 0.065 |

- **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.

Table 4 shows the results for each of our three systems. For reference, the "all runs" row shows the mean score over all systems that participated in this task. Clearly, **CLIP-YA** performs, on average, better than both of the Twitter based systems **CLIP-TW-Q** and **CLIP-TW-A**. Comparing the two Twitter-based systems, it appears that the approach that returns a tweet that is not detected to be a question (i.e., **CLIP-TW-A**), is performing better by most measures than the approach where the returned tweet is a reply to a tweet that is detected to be a question similar to the incoming question (i.e., **CLIP-TW-Q**).

In the remainder of this section, we look at the results from different perspectives.

### 3.3.1 Scores per Category

Table 5 shows the scores of our systems for each category. Health accounts for about 1/3 of all questions. This is also the category for which **CLIP-YA** performs the best. Travel is the category for which **CLIP-YA** performs the worst. But it contains less than 6% of the questions. Interestingly, this is the only category where the scores of **CLIP-YA** and **CLIP-TW-A** are somewhat comparable.

Due to a corruption in the index of the Computers & Internet category, **CLIP-TW-A** did not answer any of the questions in that category.

Table 5: Number of questions answered by each system for every category, with the corresponding score.

| | CLIP-YA | | CLIP-TW-Q | | CLIP-TW-A | |
|---|---|---|---|---|---|---|
| | Score | # | Score | # | Score | # |
| Arts & Huma. | 0.64 | 118 | 0.06 | 115 | 0.16 | 111 |
| Health | 0.77 | 327 | 0.08 | 325 | 0.20 | 293 |
| Beauty & Style | 0.50 | 119 | 0.13 | 116 | 0.21 | 107 |
| Sports | 0.51 | 122 | 0.13 | 120 | 0.23 | 115 |
| Home & Garden | 0.60 | 47 | 0.02 | 47 | 0.09 | 44 |
| Pets | 0.65 | 94 | 0.02 | 94 | 0.17 | 82 |
| Travel | 0.29 | 58 | 0.15 | 57 | 0.25 | 53 |
| Comp. & Inter. | 0.60 | 194 | 0.15 | 192 | - | 0 |

### 3.3.2 Using the Body of the Question

Our systems have different strategies to decide whether to use the terms that appear in the body of the question for retrieving answers (Section 3.1.2). For **CLIP-YA**, we delegate this decision to a classifier. This classifier chose to use the body of the question in only 11 out of 1,079 questions (Table 6). The average score over these questions (0.82) is higher than the average score over the questions where only the terms of the title were used (0.62).

Table 6: Number of questions answered by each system with(out) using the body, with the corresponding score.

| | CLIP-YA | | CLIP-TW-Q | | CLIP-TW-A | |
|---|---|---|---|---|---|---|
| | Score | # | Score | # | Score | # |
| Body used | 0.82 | 11 | 0.10 | 199 | 0.37 | 65 |
| Body not used | 0.62 | 1068 | 0.09 | 867 | 0.18 | 740 |
| - body empty | 0.50 | 387 | 0.09 | 380 | 0.20 | 267 |
| - timeout | - | 0 | 0.09 | 461 | 0.15 | 395 |
| - risk timeout | 0.68 | 681 | 0.08 | 26 | 0.19 | 98 |

For both **CLIP-TW-Q** and **CLIP-TW-A**, the answer retrieval for a substantial number of questions using the body timed out (461 or 395, respectively). In some additional cases (26 or 98, respectively), the retrieval using only the title of the question took over half of the allowed response period. These two systems do not even attempt to use the body of the question when this happens. As we have observed for **CLIP-YA**, the questions for which the body was used got an average score higher than those for which only the title was used, although for **CLIP-TW-Q** the difference is quite small.

### 3.3.3 Retrieval Field for Old Yahoo! Answers

The classifier used by the **CLIP-YA** system chooses between three configurations for the fields to be searched in the old answers. As shown in Table 7, in most cases (784 of 1,079), the decision was to match the incoming question against the content of the old answers. Questions for which this configuration was selected had an average score higher than those for which the classifier chose to search in the content of the old questions.

Table 7: Number of questions answered by CLIP-YA depending on the retrieval field, with the corresponding score.

| | CLIP-YA | |
|---|---|---|
| | Score | Count |
| Question title | 0.43 | 109 |
| Question title and body | 0.54 | 186 |
| Answer | 0.67 | 784 |

### 3.3.4 Best CLIP-YA Configuration

Combining these insights, we might speculate that the best configuration of our **CLIP-YA** system would be one

Table 8: Number of questions answered by each system for each corpus, with the corresponding score.

| | CLIP-TW-Q | | CLIP-TW-A | |
|---|---|---|---|---|
| | Score | Count | Score | Count |
| Selected tweets (small) | 0.11 | 158 | 0.46 | 24 |
| Random tweets (large) | 0.09 | 908 | 0.19 | 781 |

that uses the title and the body of the incoming question as a query (Section 3.3.2), and the index of old answers for retrieval (Section 3.3.3). As it happens, only three questions were run using both of those conditions together; their average score is 1.67. Although based on too little data for us to draw any firm conclusion, that average is certainly high enough to get our attention.

#### *3.3.5 Effect of Twitter Retrieval Corpus*

Our retrieval of answers from Twitter uses the union of two disjoint corpora, a large corpus of random tweets and a smaller focused corpus of selected tweets. For every question, we can thus look at the origin of the returned tweet (the small or the large corpus). As Table 8 shows, when an answer is found in the smaller focused corpus, the average score is higher. This suggests that a larger (i.e., longer) focused crawl of tweets that are expected to match the expected question categories might be worthwhile.

## 4. CONCLUSION

We have presented the general architecture and the implementation details for the six runs we submitted for the Microblog Real-Time Filtering task, and the three 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 Microblog systems. We are satisfied with the recall of our relevance model, but we want to focus more on precision (i.e., returning tweets only when there is high enough confidence in their relevance).

The results of the LiveQA track show that a system returning answers based on old questions and answers from Yahoo! Answers performs better on average than two systems that return tweets as answers, but that a tweet-based system may be useful for answering some difficult questions (e.g., those in the *Travel* category). In the future, we plan to tune our systems to use the body of the new questions and the content of old answers more frequently. We also plan to track the words related to the categories of interest over a longer period in order to gather a richer collection of potentially useful tweets.

## 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] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. *IEEE Transactions on Knowledge and Data Engineering*, 18(1), 2006.

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

[3] W. Magdy, W. Gao, T. Elganainy, and Z. Wei. QCRI at TREC 2014: Applying the KISS principle for the TTG task in the Microblog track. In *TREC*, 2014.

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

[5] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor. Learning from the past: Answering new questions with past answers. In *WWW*, pages 759–768, 2012.

[6] Z. Zhao and Q. Mei. Questions about questions: An empirical analysis of information needs on Twitter. In *WWW*, pages 1545–1556, 2013.