ABSTRACT

Title of dissertation: CROSS-PLATFORM QUESTION ANSWERING IN SOCIAL NETWORKING SERVICES

Mossaab Bagdouri, Doctor of Philosophy, 2017

Dissertation directed by: Professor Douglas W. Oard
College of Information Studies
and Department of Computer Science

The last two decades have made the Internet a major source for knowledge seeking. Several platforms have been developed to find answers to one's questions such as search engines and online encyclopedias. The wide adoption of social networking services has pushed the possibilities even further by giving people the opportunity to stimulate the generation of answers that are not already present on the Internet. Some of these social media services are primarily community question answering (CQA) sites, while the others have a more general audience but can also be used to ask and answer questions.

The choice of a particular platform (e.g., a CQA site, a microblogging service, or a search engine) by some user depends on several factors such as awareness of available resources and expectations from different platforms, and thus will sometimes be suboptimal. Hence, we introduce cross-platform question answering, a

framework that aims to improve our ability to satisfy complex information needs by returning answers from different platforms, including those where the question has not been originally asked.

We propose to build this core capability by defining a general architecture for designing and implementing real-time services for answering naturally occurring questions. This architecture consists of four key components: (1) real-time detection of questions, (2) a set of platforms from which answers can be returned, (3) question processing by the selected answering systems, which optionally involves question transformation when questions are answered by services that enforce differing conventions from the original source, and (4) answer presentation, including ranking, merging, and deciding whether to return the answer.

We demonstrate the feasibility of this general architecture by instantiating a restricted development version in which we collect the questions from one CQA website, one microblogging service or directly from the asker, and find answers from among some subset of those CQA and microblogging services. To enable the integration of new answering platforms in our architecture, we introduce a framework for automatic evaluation of their effectiveness.

CROSS-PLATFORM QUESTION ANSWERING IN SOCIAL NETWORKING SERVICES

by

Mossaab Bagdouri

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Dedication

To mom and dad,

Khadija, Chahid, Jannat and Ayat,

Amine, Abderrahman and Hiba.

Acknowledgments

The Prophet Muhammad, peace and blessings be upon him, said: “He who does not thank people, does not thank Allah.” (*Musnad Ahmad, Sunan At-Tirmidhî*). It has been a long journey—longer than I had anticipated or was prepared for. This is why I owe plenty of gratitude to everyone who offered an unconditional support, making a dream becoming true.

First and foremost, I would like to express my deep gratitude to Doug Oard for offering to advise me since my first day at the University of Maryland. I have been amazed by his deep knowledge, broad perspectives and continuous availability, and have enjoyed his cordial comments during our meetings at the CLIP lab, during road trips, or while he was running to catch the next flight. Doug has also opened the door for me to collaborate with incredible researchers from around the world on a diverse spectrum of projects.

I would like to thank Héctor Corrada Bravo, Hal Daumé III, Dave Lewis, and Philip Resnik for accepting the invitation to serve on my dissertation committee, and for the suggestions they provided on various occasions, including co-authorship on papers, practice talks for conferences, my thesis proposal, and of course my dissertation work. I particularly learned from Dave the practice of maintaining a detailed record of my experiments, both as a script that I can re-run in the future, and as a report that I can share with my collaborators.

William Webber deserves a special mention for getting me up-to-speed on tools and methodologies needed for conducting research on IR Evaluation, especially during my first two years at the E-Discovery lab (and this is an opportunity to acknowledge the support of the NSF E-Discovery grant No. 1065250).

I want to thank Radu Florian and Vittorio Castelli for mentoring me during an internship at the IBM T. J. Watson Research Center, where I learned several techniques that I applied in my research on question answering; and for endorsing my application for the IBM PhD Fellowship, which has supported me during my final year at UMD. It was also a pleasure to collaborate with them during the BOLT project that supported my research through DARPA Contract HR0011-12-C-0015.

A substantial portion of this dissertation was conducted in the context of the ArQAT project (QNRF grant # NPRP 6-1377-1-257). I am grateful to the valuable feedback I have constantly received from Tamer Elsayed, Walid Magdy, Mark Sanderson and Marwan Torki, and to the friendly atmosphere during my collaboration with Omid Aghili, Suliman Aladhadh, Rahma Ali, Fatimah Alqahtani, Maram Hasanain, Ahmed Mourad, Abdelrahman Shouman, and Ameer Tawfik.

I have learned a lot both from side conversations and from organized meetings with the former and current members of the CLIP lab. In this respect, I would like to thank Marine Carpuat for her patience while I was running several of my experiments reported in this dissertation on her GPUs, Jimmy Lin for introducing me to Big Data algorithms (through his course) and practice (through the Hadoop clusters), and Louiqa Raschid for her advice about my future career. I also want to thank Snigdha Chaturvedi (for telling me not to worry about leaving most of my papers out of the thesis proposal), Ahmed Elgohary (especially for introducing me to a deep-learning toolkit that I used in many experiments of this dissertation), Allyson Ettinger, Ning Gao (for performing some annotations and for coming up with a quick and effective solution that I used for evaluating one of the methods introduced in this thesis), Sergey Golitsynskiy, Raul David Guerra, He He, Hua He (notably for his patience with the heat and noise of his desktop, while I was exploiting his GPU), Mohit Iyyer (for a crash course on deep learning), Tiffany Jachja, Joseph Malionek, Khanh Nguyen, Thang Nguyen (for teaching me some basics of topic modeling and that paper that we have never submitted), Jiaul Paik, Sudha Rao, Kristine Rogers, Rashmi Sankepally (for helping out with some annotations and for the weekly meetings we had last semester, in which I pretended to be her mentor), Amr Sharaf, Bryan Toth, Ferhan Ture (for being always there when I needed some advice since he persuaded me to join UMD upon my admission), Jyothi Vinjumur (for the encouragements to pursue an academic career), Yogarshi Vyas, Yulu Wang, Ke Wu (for a crash course on machine translation toolkits), and Tan Xu (for his tips on how to maintain an excellent relationship with my advisor).

My interest in Information Retrieval started when I took an IR class from Jim Martin at the University of Colorado Boulder. I would like to thank him for advising me during the two years I spent there, and for encouraging me to apply for the PhD program at UMD. I also want to extend my sincere thanks to Leysia Palen for her amazing mentorship and personal support while I was working on my master's thesis at CU. Project EPIC was a unique learning experience as I was walking my first steps in research, inspired by the work of Ban Al-Ani, Ken Anderson, Gloria Mark, Martha Palmer, and all of my labmates.

The Tauba House in College Park has been a great community for me and my family. While all of our extended family lives overseas, I am in debt to my dearest friends and their families for their support during times of hardship, for the trust we share with each other, for the activities we performed together, and for simply allowing my kids to have somebody nearby they can consider as "uncle," "aunt," or "cousin." Al-Huda School has also been an amazing educational environment for my children. While I was overtaken by my studies and experiments—sometimes more than I should have—I was happy that Chahid and Jannat were in good hands. It is not a surprise that the presence of this school was a key factor for having decided to move to Maryland.

My endless thanks go to my mom, Salwa, for keeping me in her prayers, for her patience during the times I did not spend with her, for her hope that I'll return back soon, for, well, everything. My dad, Mehdi, has always been my role model. I cannot thank him enough for teaching me critical thinking at an early age, for the pride he shows as I advance in my career, and for the unrestricted support he has provided to me to grow my little family. I feel a great honor in seeing my brothers Amine and Abderrahman starting a career similar to mine. I thank them for this unique feeling, and I thank my little sister Hiba for the smiley faces she has been sending since she first learned how to use a smartphone.

My parents-in-law, Ahmed and Amina, my sisters-in-law, Hajar and Mariam, my cousin-in-law, Moad, and my aunt Intissar: their presence (although for short periods) was vital to several milestones along this adventure. Whether that was an out-of-state internship, an overseas conference, or simply that next deadline, I could not have made it through if it wasn't thanks to their help taking care of my family.

My warmest thanks go to my wife, Khadija. Her tremendous care and unequivocal sacrifice were not a secret. Together we set our goals, together we planned, together we traveled, but then she postponed her career so that I can establish mine. All my love and thanks go to her for reminding me of our goals when I get lost in details, for reassuring me that I'm almost there at moments of doubt, for being both a mom and a dad to our little ones, and for her hardworking days and sleepless nights. I would also like to thank my kids Chahid, Jannat, and Ayat for taking part in this journey, from its beginning, middle or end, not knowing other job for me besides being a student, and counting the days left before we go back "home."

I know I cannot return those enormous favors, but I promise to do my best to pay them forward, *in sha Allah*.

Thank you, all!

Thank you, Allah!

Contents

List of Tables

List of Figures

List of Abbreviations

Acc	Accuracy
ApD	Answer per Document
API	Application Program Interface
Aqweet	Answerable Question Tweet
BGRU	Bidirectional Gated Recurrent Unit
BLSTM	Bidirectional Long Short-Term Memory
BNS	Bi-Normal Separation
ciQA	Complex, Interactive Question Answering
CNN	Convolutional Neural Network
CQA	Community Question Answering
CV	Cross Validation
DF	Document Frequency
DNN	Deep Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HDD	Hard Disk Drive
JSON	JavaScript Object Notation
KL	Kullback–Leibler
k-NN	k-Nearest Neighbors
IDF	Inverse Document Frequency
IR	Information Retrieval
L2R	Learning to Rank
LiveQA	Live Question Answering
LSTM	Long Short-Term Memory
MAP	Mean Average Precision
MSA	Modern Standard Arabic
NDCG	Normalized Discounted Cumulative Gain
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
POS	Part of Speech
Prec	Precision
QA	Question Answering
QpD	Question per Document
Qweet	Question Tweet

Rec	Recall
RNN	Recurrent Neural Network
RT	Retweet
RTS	Real-Time Summarization
SNS	Social Networking Service
SSD	Solid-State Drive
SVM	Support Vector Machine
tanh	Hyperbolic Tangent
TF	Term Frequency
TREC	Text REtrieval Conference
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
XML	Extensible Markup Language
Y!A	Yahoo! Answers

Chapter 1: Introduction

1.1 Motivation and General Background

The Internet has for a long time been a source from which knowledge seekers can fulfill their needs. Several tools and platforms, such as search engines and online encyclopedias, have been developed to find answers to one's questions. Recent developments and wide adoption of social networking services (SNS) pushed the possibilities even further by giving people the opportunity to stimulate the generation of answers that are not already present on the Web (e.g., for novel questions), or are difficult to obtain because of different constraints (e.g., language barrier, real-time need, limitations of the device used, or lack of trust in the available information). Several online communities are built on top of different technologies to address such needs. Some mailing lists that are dedicated to specific topic or product allow subscribers to email questions and receive answers from more experts. Technical question answering platforms (e.g., StackExchange[1]) help users search in a knowledge base in addition to seeking answers to new questions. Other community driven platforms, such as Yahoo! Answers,[2] allow a wider range of topics, even those that

[1] http://stackexchange.com

[2] http://answers.yahoo.com

do not have canonical answers and are merely opinions of people who may or may not have experienced the circumstances of the inquirer. Users of more general social platforms that are based on the notions of friendship and followership such as Facebook[3] and Twitter[4] rely on their "friends" to provide trusted answers. While "friendship" motivates answering in this case, and reputation (expressed in the form of badges and points) is a key stimulus for community questions and answers (CQA) websites, financial compensation encourages workers to answer questions, presented in the form of tasks, in crowdsourcing marketplaces such as Amazon Mechanical Turk[5] and CrowdFlower.[6]

A person who asks a question somewhere typically hopes to receive an answer at the same venue. Indeed, when a user issues a query to an online search engine, she looks forward to examining the links returned by this engine just below the search box. On the other hand, when someone posts a question on a CQA website, she will revisit the question page, where answers will eventually be sent by other users. Likewise, a tweeter who posts a question on her account is likely anticipating replies from other users to her tweet.

People choose a particular platform for a wide range of reasons such as limited awareness of available resources, contrasting expectations from different platforms, low hope for a sufficient reward from a lengthy registration process, prior experience with previous questions, and technical constraints. Indeed, some recent studies

[3]http://facebook.com

[4]http://twitter.com

[5]http://mturk.com

[6]http://crowdflower.com

compared the satisfaction of users with a particular resource vs. another, as is the case for social networks vs. search engines [106], and Web search vs. CQA services [86]. Nonetheless, this selection is, perhaps, arbitrary in several cases, and might lead to suboptimal user satisfaction. Therefore, an automated system that returns answers from different platforms has the potential for improving user satisfaction.

We[7] have just introduced the problem of cross-platform question answering. It is composed of four main components, which we described next.

1.2 General Architecture

We organize the general architecture of cross-platform question answering into four components: the users and their questions, platform scoring and selection, answering, and answer presentation. We illustrate them with Figure 1.1.

1.2.1 Questions

The entry to our architecture is an information need expressed by a user who might be willing to trade timeliness for accuracy. This information need can reach our system in several ways. But we focus in this dissertation on three different scenarios.

In the first scenario, a user has a general interest in some broad topic, and wants to stay up to date on it. A motivating use case is a brand management specialist who needs a real-time tool for monitoring the perception of her brand by

[7]Using 'we' in this dissertation is a style choice that refers to the author, unless there is a pointer to a publication of two or more co-authors.

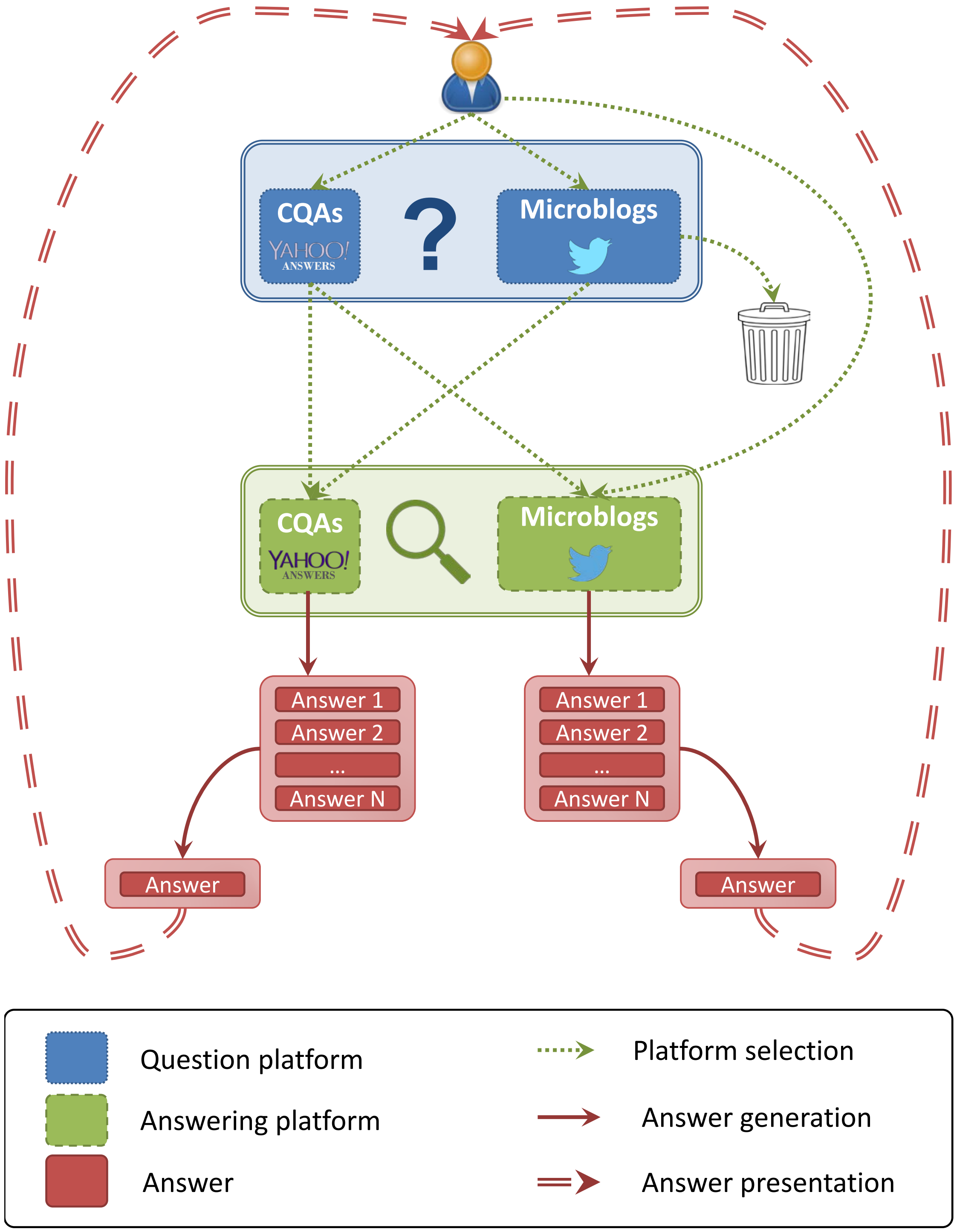


Figure 1.1: Implementation of the cross-platform question answering architecture.

its customers. This specialist would prefer receiving a notification as soon as some important event is detected, such as an expression of dissatisfaction by unhappy customers, rather than performing a periodic search every now and then. However, the same event might be reported several times (e.g., disruption of service in some region), and receiving a notification for each occurrence (even if it was expressed in different ways) might do more harm than good, as it can cause distraction from paying attention to other important events. Thus, a notification should be sent only once for each critical event (potentially with a weight indicating its importance). As the search is expected to have continuing value to the user over an extended period (i.e., the information need is expressed only once, but the notifications will be sent for a long period), we assume that the user provides extensive details about her topic of interest, instead of just a couple of terms as is typically practiced with search engines. Those details represent at the same time an opportunity for scoping the exact information need, and a challenge for automatically detecting important from less important pieces of information. In this dissertation, we study a variant of this use case in which a user has a general information need, and the stream of documents from which the notifications generate is Twitter's public stream.

In the second scenario, the user has a specific information need, optionally about some personal experience, expressed as a complex question with extensive details to a community of people. This user hopes that someone from the community will read the question and then will provide a useful answer (e.g. a fact or an opinion) within a reasonable period of time. But she would also consider an automatically generated answer if it is made available spontaneously. An automated system would hence need to analyze the question, find some related old answers, construct one answer from them, and present it to the asker. While there is a similarity with the first scenario in terms of the opportunity and the challenge present with the detailed question, there are also several differences. First, the system has to acquire a large repository of answers; and given typical resource limitations (e.g., crawling time and disk space) and some expectation of the topics of the questions to be posted, it has to focus on collecting content that has a high chance of being useful for the future (i.e., unseen) questions. Second, the question here is expected to have a narrow focus, compared to the broad topic of interest in the first scenario. Third, the question is written with an expectation that a human is the primary reader, while in the first scenario the user is aware that she is communicating with a machine.

In the third scenario, the question occurs naturally intermingled with other content. This arises often in online interaction with help desk, and a bot that can proactively detect questions and provide answers (e.g., by searching in a knowledge base) will expedite solving the problem that the customer is facing. However, automatically detecting that something is a question has its own challenges. In fact, question marks and cue phrases are not perfect indicators of questions with real information needs. Some questions, for instance, are rhetorical (e.g., imagine a bot trying to answer the question "how is your day going so far?"), and we should not answer them. Even those that convey a real information need might be unanswerable due to missing context or vital details. Hence, we can save some processing time (and our face) by not attempting to answer them.

1.2.2 Answering Platforms

Once some question is detected, either explicitly by an indication from the user or implicitly as a prediction of a classifier, we would like to provide an answer. In this dissertation, we only examine returning answers by searching in one or more corpora, and defer other alternatives, such as routing the questions to experts (e.g., see our own work about finding journalists in Twitter [7]), to future work. Within the many platforms available for searching for answers, we study two SNS families, which are microblogs and CQA services.

- **Microblogs:** Microblogging services, such as Twitter, enable a rich communication behavior with various usage patterns. A prevalent practice, among others, is asking questions. It is natural, thus, to consider Twitter as a platform for answering at least some types of questions. To do that, we need to build a classifier that detects questions that are seeking real answers, and another that detects which tweets are useful for providing answers.

- **CQA services:** Community question answering websites are getting more attraction recently. Some of them target a community specialized in some particular field, such as the technical platform StackExchange. Others, such as Yahoo! Answers, allow a wider range of topics, even those that do not have canonical answers and are merely opinions of people who may (or may not) have experienced the circumstances of the inquirer. In this work, we focus on Yahoo! Answers, which is a well established platform that has been in service for over a decade. It contains hundreds of millions of questions with their answers in one main and 22 localized versions.

Once a platform is selected, there are different options for searching there. One way is online, such as by using the internal search engine of that platform. This is an attractive option, since substantial efforts might have been put in by the developers of that platform to tune their search results. However, the use cases and structure of questions on which that internal search engine was optimized might be different from ours. Thus, our strategy in this dissertation is to collect a large amount of data from the selected platform, whenever applicable, to use it as a primary source for searching for answers; and to rely on its search engine only as a secondary source for enriching our pool of answers. By doing so, we gain more control on the parameters we want to study, but we also collect a substantial amount of data that can be used to train a classifier (or re-ranker) to enhance the performance of our retrieval.

1.2.3 Answering

Given a question and an answering platform, we might need to perform some sort of question transformation from the source to the target platform. In fact, different platforms adopt different conventions for asking questions. In microblogging services, there usually is a limit on the length of the post (e.g., 140 characters for Twitter). Users can also specify a hashtag that indicates that the post is actually a question, which would increase its exposure to candidate answerers. Another convention is to seek the attention of certain user (e.g., an expert in the topic of the question) by mentioning her username. On Yahoo! Answers (Y!A) and other CQA services, there is some structure associated with each question, such as organizing it into a title and a description, and assigning it to a category. The particularities of how the questions are represented in different platforms need to be accounted for while we try to find useful answers.

The content of the question is not the only source of signal for considering what to retrieve. Additional evidence can be available as a set of features from the asker, such as her history of previous questions and her interaction with other users. However, we neglect asker information in this dissertation, deferring personalization to future work.

1.2.4 Answer Presentation

The available answers can be presented in different ways to the asker. For instance, on a web browser, we can have a ranked list for each selected platform in a different tab. In addition, we can merge the answers from the various answering platforms and present them as a single combined list. We can also return a single answer, which can be selected from available ones or synthesized from their content. Another way to present an answer is through a push notification into a smartphone.

1.3 Evaluation of Answering Platforms

Building good answering systems requires the ability to evaluate their performance. Ideally, we would like to base our evaluation on direct feedback from the users of those systems. But due to the complexity and cost of user-centric evaluation (e.g., we need ground truth labels from a large and representative set of users whose perception of relevance might be different), we adopt a system-centered evaluation that exploits the annotations of independent trained assessors who indicate the usefulness of the retrieved answers.

We adopt three evaluation approaches in this dissertation. Our first (and preferred) choice is a direct evaluation, by a third party, of the answers returned by our answering platform. For over two decades, the National Institute of Standards and Technology (NIST) has been organizing several evaluation campaigns in the context of the Text REtrieval Conference (TREC). Every year, a set of tracks are introduced about some specific themes in information retrieval, and teams around the world (interested in those themes) compete by building the best systems they can to solve the problem addressed in those tracks. In each of the past two years, we participated in the Live Question Answering (LiveQA) track; and in 2016, we participated in the first edition of the Real-Time Summarization (RTS) track. The organizers of these tracks have provided us with useful annotations for evaluating our answering systems.

Some of the problems we study in this dissertation have unique characteristics, and have not been addressed by any evaluation campaign of which we are aware. In such cases, we collect the annotations needed for our studies by referring to crowdsourcing. In this second evaluation approach, we generate the output of several systems we want to compare, and ask independent paid workers to assess the quality of this output, which allows us to draw some conclusions. The confidence in the annotations obtained with this approach is typically lower than that of the first one because (1) there is less control on the training of the annotators, and (2) there might be some experimenter bias, since the same person who is conducting the research is also managing the annotation process. However, this evaluation approach has three benefits, as (1) it allows us to study the actual problem of interest (instead of a variant that complies with TREC's setup); (2) it provides a quick feedback about the issue we are examining (compare a few hours or days in crowdsourcing to several months in typical TREC evaluation); and (3) it has relatively a low cost (a few hundred dollars for the entire problem we study).

Our third approach is an instance of indirect evaluation. The cost associated with either of the previous evaluation approaches can be amortized if the annotations that have already been collected are reused to evaluate new systems (i.e., those that did not produce the original answers that were assessed). Test collection reusability has been studied extensively in the context of closed corpora (i.e., when there is a finite pre-determined set of documents that are used for retrieval). For our goal of studying the usefulness of new answering platforms, we will typically have a new set of documents, and traditional evaluation techniques with incomplete judgments cannot be applied in such cases. Hence, it is our goal in this dissertation to also study ways of estimating the effectiveness of new answering platforms when none of their answers have been annotated before.

1.4 Research Questions

This dissertation argues that **cross-platform question answering extends the ability to satisfy the information needs.** We support this claim by addressing the following research questions:

RQ1. How well can we detect answerable questions in Twitter?

RQ2. How well can questions asked in one platform be answered using information from another one?

RQ3. How well can automated question transformations improve the ability of automated matching systems to answer Twitter questions from Yahoo! Answers?

RQ4. How well can we make a binary decision whether to return an answer?

RQ5. How well can we estimate the effectiveness of a new answering platform before any of the answers from that platform have been manually scored?

1.5 Contributions

Answering the research questions listed in Section 1.4 results in several contributions that are distributed across generalizable knowledge (K), classification and retrieval systems (S) for which we advance the state of the art for existing tasks or examine the effectiveness for novel tasks, framework and metrics for evaluating (E) some of those systems, and corpora (C) for further investigation of our novel tasks.

K1 We introduce the general problem of cross-platform question answering (Section 1.2).

S2 We advance the state of the art for detecting questions in microblogs that convey a real information need (Section 3.2.4.2).

S3 We introduce the problem of identifying answerable questions in Twitter (aqweets) and study the performance of two families of classifiers for solving it (Section 3.3).

K4 We introduce an end-to-end pipeline for detecting aqweets, and examine the performance of its main components (Section 3.4).

C5 We release a test collection for aqweet detection (Section 3.3.2).

S6 We study an instance of making a binary decision on whether to return an answer from an answering platform given that the asker might have already received some replies to her question from her friends (Sections 4.3.7 and 4.3.10).

C7 We release a test collection for answering aqweets using Yahoo! Answers (Section 4.3.9).

K8 We develop a method for collecting tweets that are likely to contain rich information relevant to some category of questions (Sections 4.2.2.2 and 4.4.3.7).

S9 We suggest two different ways for using tweets to return answers to Y!A questions (Section 4.2.2.3), and compare their effectiveness (Section 4.4.3.2).

S10 We build a state-of-the-art system for answering new Y!A questions using a large crawl of Yahoo! Answers (Section 4.4.1.2).

K11 We introduce a method for adapting the training of the TREC Microblog data to build a learning-to-rank model for scoring candidate tweets to be returned as answers for the TREC LiveQA track (Section 4.4.2.2).

K12 We compare the effectiveness of two answering platforms on different categories of questions asked on Yahoo! Answers (Section 4.4.3.3) and study the potential gain from combining their answers (Section 4.4.4).

K13 We provide a simple, but effective, function that correlates the size of a category of questions and their diversity with the performance of the state-of-the-art answering system on that category (Section 4.4.3.3).

S14 We investigate the problem of making a binary decision for whether to return relevant novel tweets, and develop a corresponding system that has a state-of-the-art effectiveness (Section 4.5).

E15 We introduce a framework for evaluating automatic performance estimators of systems that retrieve only new answers (Section 5.2).

E16 We introduce a new measure for automatic estimation of the performance of systems that return only new documents, and study its effectiveness on the TREC 2015 LiveQA and TREC-8 Ad Hoc tracks (Chapter 5).

E17 We contribute to the design of a new correlation coefficient that focuses on the maintenance of score differences between highly ranked ablated systems (Section 5.2.1).

1.6 Dissertation Outline

The remainder of this dissertation is structured as follows: We survey related work in Chapter 2. In Chapter 3, we study the detection of answerable questions in microblogs. Next in Chapter 4, we examine several cases of cross-platform question answering, including answering Twitter questions using content from Yahoo! Answers, answering questions from Yahoo! Answers using content from both Twitter and Yahoo! Answers, and deciding whether or not to return a potential response. Then in Chapter 5, we introduce an automatic evaluation measure for assessing future answers, which makes it possible to estimate the effectiveness of new answering platforms without new annotations. We conclude in Chapter 6, scoping our contributions in the context of experimental limitations, looking to future directions, and articulating some of the broader impacts of our work.

Chapter 2: Related Work

The work reported in this dissertation is related to several lines of research. Our work is akin to early efforts in factoid question answering (Section 2.1), notably on their focus on the reusability of complex, interactive QA test collections. The choice of a particular answering platform was faced by all participants of the TREC LiveQA track (Section 2.2). Chapter 5 about estimating the effectiveness of new answering platforms is an instance of IR evaluation, and is complimentary to previous approaches for evaluation with incomplete judgments (Section 2.3). Looking for similar past questions to answer new questions is inspired by findings from research on community question answering (Section 2.4). The cross-platform notion is analogous to some instances of federated search (Section 2.5), and shares some similarities with community question routing (Section 2.6), especially in terms of proactive intervention by systems to increase the likelihood of satisfying askers' information needs. We focus this chapter on describing the relevant related work on those topics, but we note that more detailed pointers to some related work on the question detection and the TREC LiveQA and RTS tracks can be found in Chapters 3 and 4, respectively.

2.1 Factoid Question Answering

The first key effort in modern QA is, perhaps, the TREC-8 Question Answering track [136]. The motivation was to shift traditional information retrieval systems from merely indicating which documents contain an answer, to returning the answer itself. Since then, a wide range of approaches have been proposed and a large number of applications have been developed. QA systems usually contain three processing steps [65], starting from query formulation and classification, going to document and passage retrieval and finishing by answer processing. Such systems assume the existence of a corpus of documents containing the answer, and the engine is responsible for finding it.

The first stage, question processing, aims to enrich the question with sufficient context to infer the intention behind it. Such processing might involve using information about the user, considering previous questions in the same session, exploiting the domain knowledge for the inferred question type, and making use of linguistic resources such as part-of-speech tagging, WordNet [100] and PropBank [71], especially for the question. Harabagiu [53] indicates that this step is important for both finding a candidate set of answers, and selecting an answer out of them.

Extracting an answer often involves deep Natural Language Processing (NLP) computation that makes the offline pre-processing for a large collection prohibitively expensive [51]. To mitigate this challenge, the question, or a rewritten version of it, is issued as a query to an index of the collection of documents where answers might exist, and returns "hotspots," which are text fragments that have high relevance to the topic of the question [30]. Complex linguistic processing, including answer extraction, is then applied only to the retrieved hotspots.

Following the success of the early TREC QA tracks, another series of tasks was introduced under the name complex, interactive Question Answering (ciQA) [66]. These questions still seek answers about facts, but are more complex in their nature. For instance they look for the definition of a concept or the relationship between several entities. Interestingly, this task stimulated a wide body of work on the (re)usability of test collections created within the context of such tasks. Lin and Demner-Fushman [81, 82] introduced POURPRE as a unigram metric that exploits a list of "information nuggets" to evaluate definitional questions. It simply counts the overlap between the answer and each nugget after ensuring that all words appear within the same answer string. They find a high correlation between the scores computed using this metric and the official scores provided in the ciQA tracks of TREC 2003-2005. In a subsequent work, Marton and Radul [95] proposed the metric Nuggeteer, which gives high credit to n-grams that appear in a large number of nuggets, and scores new answers accordingly. Both lines of work show that the proposed metrics experience a high correlation with human assessments, when compared to other n-gram metrics such as ROUGE [80] and BLEU [111]. Nevertheless, the corpus for these experiments is limited to the news domain, and there is no indication of whether the suggested metric is appropriate for evaluating runs that did not participate in the pool of answers that was used to create the nuggets.

Nugget-based evaluation became since a standard for assessing systems answering complex questions [83], with a body of work analyzing the stability [83] and reusability [113] of test-collections when nuggets are manually prepared by human assessors. Other pieces of work focused on the extraction of nuggets from documents for evaluation purposes in a semi-automated manner [41, 118, 157]. These efforts were also limited to the context of factoid questions (including complex questions) and do not provide recommendations for fully automated approaches when nuggets are not already present in a test collection.

2.2 Cross-Platform Live Question Answering

A substantial amount of this dissertation work was conducted during our participation in the TREC LiveQA track [1, 2]. We provide additional details about this track in Section 4.1.1, but we mention here that this challenge was characterized by a stream of live questions posted on Yahoo! Answers, and that the participating systems had to return a short answer within one minute from whatever online or offline resources they could assemble. Hence, in addition to the traditional challenges of providing a good scoring function (e.g., by applying a state-of-the-art IR ranking function, or by training a learning-to-rank classifier), the participants had to decide, first, which platform should be interrogated for generating candidate answers. In some instances, they also had to face the issue of selecting a final answer from different platforms.

All of the participants (to the best of our knowledge) used some content from at least one CQA website. With the except of Marx and Coelho who indexed a crawl of StackOverflow and used it for retrieving the answers of some questions

[96], the other teams used Yahoo! Answers, sometimes in combination with up to seven other CQA services. The strategies for accessing and using Y!A differed across participants. One approach relied on using the internal search engine of Yahoo! Answers [3, 32, 68, 104, 122, 123, 140, 141, 145, 155]. In a second approach, participating systems used the Webscope L6 corpus [129] (which was collected in 2007 and contains over 4.4 million questions from Y!A with their answers) as training data for answer ranking [122, 135, 140, 155], as an indexed source of retrieved answers [15, 28, 36, 92, 94, 130], and/or as a way to estimate term statistics, such as document frequencies [140].

Several systems used Web search as a source of candidate answers. Bing was a popular choice thanks to its API[1] (e.g., [32, 68, 92, 94, 122, 123, 135, 139, 140, 141, 145]). But a couple of other systems relied on Google [94, 104], and in one instance [36], on Google Knowledge Graph.[2]

Combining answers from different sources was a problem that was addressed in a variety of ways. Some systems were agnostic to the platform from which the answer originated. They implemented such an approach either by merging different corpora in a single local index [28, 92] or by retrieving a set of candidate answers from different platforms before applying the same scoring function to all of them [68, 94, 96, 122, 123, 140, 141]. In either case, some choices had to be made about how to represent the documents, such as by having a short field that corresponds to the title of an old CQA question or the title of a web page, and a long field that

[1]https://www.microsoft.com/cognitive-services/en-us/bing-web-search-api

[2]https://developers.google.com/knowledge-graph/

corresponds to the body of the old CQA question or the snippet of a Web search result. Another parallel approach involved a manually set weight for the answering platforms incorporated in the scoring function [3, 145], with higher weights set for the CQA services compared to search engines, and a higher weight configured for Yahoo! Answers compared to other CQA websites. Yet another approach was a strategy in which the answering platforms were interrogated in a sequential order, moving from one platform to another only when a small number of candidate answers were retrieved [32] or when a score threshold was not met [36]. In some cases, the participants submitted systems retrieving answers from different platforms, without attempting to merge them [92]. That is, they had an independent system for each platform.

Our systems are different from the approaches above in two major aspects. First, we participated with two systems that each retrieve answers from a different type of platforms, namely microblogs, which is Twitter. Second, for our third system, we crawled a large corpus of questions and answers from Yahoo! Answers, which increased our chances of finding answers (compared to using Webscope L6), allowed us to control the scoring of the candidate answers (compared to using Y!A's internal search engine), and provided us a substantial amount of training data. We shared a portion of our large crawl with Malhas et al., and their results show that their systems that included that crawl performed better than their other system that did not include it [94].

2.3 Evaluation of Information Retrieval Systems

Information Retrieval is inherently an empirical discipline. As a consequence, evaluation has been one of its central foci, allowing us to compare between different retrieval techniques and to track our progress in finding useful content. Evaluation of IR systems is typically based on four dimensions: an information need, an IR system, an item retrieved by the system for that information need, and an assessment of the usefulness of the retrieved item for the information need. Different applications call for different ways of combining these dimensions. For instance, we might be interested in a single information need (e.g., finding privileged documents in an attorney-client communication [31, 105]), a pool of queries within one broad subject (e.g., finding scientific abstracts relevant to available medical conditions [120]), or a large sample of diverse topics (e.g., in Web search). Our goal might be to improve an existing single system (e.g., an internal enterprise search engine), study different variants of an algorithm (e.g., with parameter tuning), compare a diverse set of independent systems (e.g., in a competition), or even examine the ability of human "systems" to find relevant information (e.g., to establish a reasonable effectiveness target for automated systems). The user might have an interest in a single useful piece of information (e.g., the address of some place), an exhaustive enumeration of all relevant materials (e.g., with patent search [91]), or something in between (e.g., recommendations of nearby restaurants). Finally, the assessments can be made by the user who formulated the query either directly (e.g., by providing a score) or indirectly (e.g., through a click [56]); by a third party that can be a single person or a committee of assessors who might be experts in the topic of the information need (e.g., medical doctors or senior lawyers), untrained workers (e.g., in some instances of crowdsourcing [75]), or non-expert trained annotators (e.g., undergraduate students in a controlled experiment); or by another automated system that has access to some information (e.g., the ground truth of a sample of items) that is not available to the retrieval system being assessed.

The primary goal of IR evaluation is to produce an unbiased and sufficiently precise estimate of the difference in the effectiveness between different system configurations (e.g., ranking techniques, sources of information, etc.) for future data (i.e., new systems, queries or documents). This estimation can be improved by reducing the variance in measurement, which typically translates into acquiring more information (e.g., more assessments); or increasing the quality of measurement (e.g., by training the annotators). But that might come at an unaffordable cost. For this reason, research in IR evaluation often examines ways for reducing the measurement cost while maintaining the ability to detect statistically significant differences. For instance, observing that the average precision follows a normal distribution [25], Carterette and Allan took advantage of the cluster hypothesis [119] to show that inter-document similarity can be useful to minimize the number of judgments needed for comparing the mean average precision [18] of different retrieval systems [24]. In a binary relevance setup, they used cosine similarity to estimate the probability of relevance of unassessed documents given a prior distribution over their relevance, and the labels of assessed documents. This approach is useful to minimize the number of annotations by propagating the relevance judgments through the pooled documents. This approach, unfortunately, does not generalize to cases where new systems are present with only brand new documents.

Büttcher et al. [21] studied the bias in evaluating systems that did not contribute to the pool of assessed documents within the context of the TREC 2006 Terabyte track [20]. They showed that some standard IR evaluation measures react in opposite directions, as bpref [19], for instance, appears to be favoring such systems, while average precision is penalizing them. Then, they demonstrated that a slightly more stable evaluation can be obtained with two classifiers (support vector machines and Kullback-Leibler divergence) trained on the pool of assessed documents to predict the relevance of the unassessed documents. However, there is no indication of the usefulness of the assessed documents to compare two non-participating systems, and this work is also limited to binary relevance.

2.4 Community Question Answering

With the emergence of Web 2.0 and Social Media, community driven question answering platforms became popular. The research community followed up by suggesting approaches to answer new questions from old answers or to route the questions to expert users (Section 2.6). Wang et al. [142] focused on finding similar questions to a new question based on matching syntactic tree kernels. They obtained good performance by returning the answer of the most similar question. A simpler model was proposed by Shtok et al. [125]. They answered new questions from past answers by first selecting candidate similar questions with cosine similarity, and then training a classifier that predicts the performance of a past answer given the new and old questions. Zhou et al. [158] took the approach of expert finding to map questions to the users who might answer them. In a supervised setup, they showed that KL-divergence between the new question and the language model of a user's answered questions yields good results when independent statistics of the categories of the new question and past answers are included as features. Nevertheless, Yen-iterzi and Callan pointed out to a potential bias that arises from using the votes of old answers as ground truth for evaluating such systems [148].

2.5 Federated Search

Our work can been seen as an instance of federated search [124]. In this problem, a system is presented with a query, and needs to match it against several resources, returning documents from each, and combining them together in some presentation layout. Some motivations of this work can be traced back to the early days of Web search, when each search engine was indexing only a fraction of the Web [13]. In such a case, a metasearch engine obtains results in an online manner from other search engines before combining them [98]. An example of a metasearch engine would be a portal that shows, for a given query, results from Bing and Google. A more recent instance of federated search is known as aggregated search [4]. In this case, the engine has a number of indices, one per vertical, and runs the query on a subset of them, before moving to the combination step. Most modern web search engines have this functionality built in. For instance, when searching for

"Paris," Google shows a map, a definition box, and a box for pictures, in addition to "normal" search results. These, perhaps, correspond to four verticals.

Most of the research on federated search focuses on three aspects: representation, selection and merging [124]. Collection representation deals with gathering and maintaining information about the individual search engines or verticals, such as the terms statistics [50] and query logs [5]. In uncooperative cases (i.e., the individual collections do not expose their information), statistics can be approximated from documents returned after issuing a series of queries [23]. To optimize the usage of resources, a selection phase is often performed to restrict the collections that will be searched for a particular query. For this, a similarity is calculated between the query and the collections. For instance, the GIOSS method gives preference for collections that are estimated to have a high number of documents containing all of the query terms [49]. On the other hand, the CORI algorithm adapts the BM25 model to estimate the "belief" in associating a term with a collection [22]. Finally, the search results are returned from selected collections, and need to be merged. The main challenge in this stage is to convert collections-specific retrieval scores into global scores. For example, Si and Callan [126] proposed to train a regression model to predict this score mapping. Learning takes place by creating a large index that contains a sample of documents from the individual collections, and then comparing, for a given query and a matching document, the local and global retrieval scores. Other papers addressed finding duplicate documents across different collections (e.g. [154]).

2.6 Routing Questions to Users and Communities

A wide body of work is interested in matching questions with experts in CQA services and other communities. For instance, Pal and Konstan [108] observed that experts in a tax-related CQA tend to prefer answering questions that have not yet received good answers. This question selection bias is then exploited for early identification of experts in such a community [109]. Pal and Counts examined methods for finding authoritative accounts on Twitter for a particular topic [107]. They showed that probabilistic clustering is useful to eliminate outliers, and that ranking users with a Gaussian algorithm is both effective and efficient. Looking at the problem from the opposite perspective (i.e., the expert is already available and wishes to receive questions), Dror et al. [37] modeled question routing as a recommender system that suggests to potential answerers questions in which they might be interested. Among several classifiers, they found that a Gradient Boosted Decision Tree [43] was the most robust for recommending new questions in Yahoo! Answers to candidate users. They also noted that content features have better prediction power than social features, and that better performance can be obtained by combining both.

Instead of relying only on individual experts, sometimes a group of people may be better positioned to provide an answer in a collaborative fashion. Chang and Pal [27], for instance, indicated that questions for which there are several contributors are more likely to experience a long lasting value (measured as the total number of views) than those with only one answerer. Bouguessa et al. [16] introduced the concept of a knowledge-sharing community, which is composed of askers and authoritative users, all interested in a particular topic. They used the Expectation-Maximization algorithm to estimate the parameters of the proposed mixture model, and identify the authoritative users within these communities. Pal et al. [110] suggested that a question could be routed to a community of users. They studied question, user and community features, in addition to some similarity metrics among them, which they used to propose a k-NN based algorithm for routing an incoming question.

Routing questions between heterogeneous platforms has seen little work. Oeldorf-Hirsch et al. [106] built a simple interface that contained a text zone for asking questions and four options to direct the question to a one of two social networking services or one of two search engines. A total of 82 recruited participants used this interface to ask questions and were given a few days before collecting the answers. The results showed that most questions were directed to search engines, especially for prompted information needs. Opinions and recommendations are the questions the mostly asked on social media for unprompted information needs. Jeong et al. designed and implemented a crowdsourcing pipeline to answer questions asked on Twitter [59]. They found no difference in quality between the answers of crowdworkers and those of the friends.

The next chapter explores the first component of our framework: detecting answerable questions.

Chapter 3: Detecting Answerable Questions[1]

Most research on answering questions has assumed that questions are easy to recognize. For example, in the TREC Question Answering track [136], questions were found in the <question> field, and in the more recent TREC LiveQA track [1, 2], real people type real questions into a text box that is followed by a button labeled “Find Answers.” In many potential applications of question answering systems, finding the question is not so straightforward. For example, help desk staff might resolve problems more quickly if automated systems were to listen to an interaction and automatically bring up possible answers to a user’s question that the agent could then select from and interpret on the fly [101]. Today’s automated assistants such as Siri and Cortana respond only when they hear a clue phrase that indicates a question is coming next, but one could imagine a brave new world in which evolved versions of such systems help out when someone who has taken a wrong turn shouts “Darn, what should I do now?” Those systems would necessarily require a component for automatic identification of questions.

We focus on detecting questions that are asked in Twitter. As with others who

[1]Some parts of this chapter were taken from a paper in preparation by Bagdouri and Oard [11].

have worked on this task, we start by selecting questions that have some obvious marker that we would expect to be associated with questions (in our case, a question mark or phrase). Then, we attempt to automatically determine which of those are questions for which answers are desired. Not all of the questions are actually seeking an answer. In fact, some interrogative tweets are, for instance, rhetorical. Others are followed by an answer provided by the asker herself. And a number of tweets containing questions are simply advertising a product. We refer to tweets that ask questions for which an answer is expected as qweets [79].

As it turns out, merely detecting a qweet accurately is not a sufficient requirement before attempting to provide an answer. Consider for instance when someone asks her friend: "@user hey, when u coming back?" This is, indeed, a question seeking a real answer—justifying its positive label in the corpus released by Zhao and Mei [156]. Nevertheless, until we develop some very smart agents capable of inferring people's future plans from publicly accessible data, we believe that only that particular @user (or perhaps some of her friends) can provide a useful answer. Hence, we go one step further in the types of qweets we want to find, and introduce the novel problem of detecting answerable qweets, or simply aqweets. We define an aqweet as a qweet for which there might exist some stranger (i.e., someone who does not know the asker and has no interaction with her) who could potentially read the question and provide a useful answer.

The task of detecting questions in Twitter arises in several academic settings, such as when seeking to study the needs of specific populations such as scholars [117], journalists [55], or people responding to natural disasters [117]. It also arises as a component in larger systems. For instance, Paul et al. [112] found that 81.3% of the questions asked on Twitter that are not addressed to a specific user receive no response at all. With an accurate aqweet detector, one might develop automatic agents that can answer those questions or route them to a relevant expert [78].

Question detection is a classification task that has been traditionally approached by selecting some classifier design (e.g., SVM) and then experimenting with different techniques for feature selection and shaping in order to optimize some evaluation measure (e.g., accuracy or F_1). Recently, deep learning techniques have offered the promise of using generic neural network designs that are able to learn to optimally select and shape "raw" features (i.e., features that are provided in whatever form they are directly observed). We are not aware of any prior application of these techniques to question detection in Twitter, so our goal is to empirically determine whether they live up to their hype for this task. Our contributions in this chapter are, hence: (1) we present a pipeline for detecting answerable questions in Twitter; (2) we improve the state of the art on the problem of qweet detection; (3) we introduce the novel problem of aqweet identification; (4) we release a corpus that enables the study of this problem; and (5) we report the effectiveness of different classifiers and features, establishing a baseline for future work.

In the remainder of this chapter, we present the problem of finding interrogative tweets (Section 3.1), we address the task of detecting questions with real information needs (Section 3.2), and we introduce the problem of identifying answerable questions (Section 3.3). An end-to-end pipeline is proposed and discussed in Section 3.4. We summarize our findings in Section 3.5.

3.1 Detection of Interrogative Tweets (Itweets)

In a survey of 624 employees from a large company, in which they were requested to provide examples of questions they had asked on Facebook and Twitter, Morris et al. indicated that 81.5% of the examples explicitly included a question mark [103]. That is, the question mark had a recall of 0.815 in recovering posts with information needs within this small demographic. By design, this process of collecting questions has perfect precision.

The work of Li et al. provides a close recall estimate [79]. Two raters labeled (individually, and then converged after a discussion) a set of 2,045 English tweets that were published through Twitter's sample API over a period of one hour. They found that 84.6% of the interrogative tweets (defined as those that contain a question sentence) also have a question mark, and that including tweets with 5W1H terms[2] increases the coverage to 97.3%, but at the same time it drops the precision from 0.865[3] to 0.547. The authors applied two heuristics consisting of (1) requiring the 5W1H words to appear at the beginning of a sentence, and (2) adding some auxiliary words, such as "is" and "are" after "what." These heuristics helped attaining a recall of 0.907 and a precision of 0.954, and no supervised technique was able to match those values.

[2]The actual terms were not explicitly mentioned in that work, but they typically refer to What, Who, Where, When, Why, and How, as indicated in `https://en.wikipedia.org/wiki/Five_Ws`.

[3]The paper indicates the precision to be 0.969, but we found this number inconsistent with Tables 2 and 3. We confirmed, in a private communication with the authors, that the precision should have been reported as 0.865

As an alternative to the 5W1H question terms, Efron and Winget included tweets that contain any instance of a small set of phrases, such as "I'm looking for," within the interrogative tweets [40]. While they did not indicate the recall resulting in this approach, a verification over a small sample of 100 tweets shows a precision of 0.93.

All of the work above focused on English tweets, which might have some characteristics that are not generalizable to other languages. Hasanain et al. also worked on the problem of question detection, but for Arabic tweets [54]. Arabic is arguably more challenging due to its morphological complexity and to its presence as a mixture of Modern Standard Arabic (MSA) and several local dialects [33]. A notable consequence is the need for a large set of question terms and phrases to find itweets. Hence, Hasanain et al. constructed a list of 488 such phrases that they used, combined with the Arabic and Latin variants of the question mark, to detect the itweets. They reported that this method has a precision of 0.79, but they did not measure its recall.

Overall it appears that relying on an obvious marker (e.g., question mark or phrase) is a widely adopted approach in prior work that focuses on the detection or analysis of the information needs of Twitter users, and that a recall and a precision of about 0.8 are reasonable estimates. We follow this trend in the subsequent stages as well, adopting the same particular marker that had been used in the released corpora to maintain the comparability of results. That is, we use question marks for English tweets to compare our work with that of Zhao and Mei (Section 3.2.1), and question phrases for Arabic tweets for the comparison with Hasanain et al..

3.2 Detecting Questions with Real Information Needs (Qweets)

Different families of taxonomies have been introduced for categorizing the questions that people post on social media in general and Twitter in particular. Perhaps the most dominating ones are topical classification and purpose-oriented classification. Among the former classification, Morris et al. enumerated nine topics, among which Technology and Entertainment are the most frequent in their survey [103]. Using the same coding scheme, Paul et al. [112] found a different distribution in their sample of tweets where the most prominent topics are Entertainment and Personal & Health. A more fine-grained taxonomy of topics was introduced by Liu and Jansen based on an automated tagging by the OpenCalais service,[4] where Human Interest and Entertainment Culture were detected to be the most prevalent topics [88].

The purpose-oriented taxonomies, which are of interest to this dissertation, have been introduced in different bodies of work. Among these types of questions, we find primarily:

1. those that express an information need [158], which can be subjective [89] such as opinions [40, 103] and recommendations [103] or objective [89] such as factual information [40, 103];

2. those that seek help [79] or a favor [103], such as inviting [40] or coordinating [40, 103] an action; and

[4]http://www.opencalais.com

3. those that aim for some sort of dissemination, such as providing an offer that could be social [103] or promotional [79], posting a question with its answer [40, 79], expressing an opinion in a rhetorical way [40, 79, 103], maintaining a social connection [103], or simply quoting the question of another person or article [79].

Within the questions that seek to address an information need, Hasanain et al. had a more detailed look at the special demographic of Arab journalists [55]. They suggested three sub-categories for factual information needs (Find Fact, Find Information Source, and Confirm Fact), and two sub-categories for opinion-based information needs (Find Opinion, and Clarify Opinion).

To operationalize the types of questions as a classification task, a binary label indicating whether a tweet is a “qweet” has been suggested. However, this term, coined by Li et al. [79], appears to have slightly different definitions among researchers. In fact, a qweet was originally referring to tweets that “ask for some information or help” [79], suggesting to exclude tweets of type (3) and include those of both types (1) and (2). On the other hand, Hasanain et al. mean by qweets the tweets that “convey real information needs” [54], which is what Zhao and Mei also focus on, but without using the term qweet [156]. With this definition, tweets of type (2) are not qweets. In this dissertation, we aim to find questions that convey real information needs, and hence we use the term qweet to refer to the latter definition. This section studies the detection of qweets among itweets.

3.2.1 Qweets Corpora

We study the qweet detection problem in two corpora. For English, we use the collection released by Zhao and Mei [156]. They released 2,466 tweets of the 3,119 that had been used for the experiments they reported, so as a baseline we replicate their study on those 2,466 tweets.[5] Because their collection is balanced, they (and we) report accuracy as an evaluation measure. For Arabic, we use the test collection released by Hasanain et al. [54]. They released 3,341 of the 3,342 tweets that had been used for the experiments reported in their paper (one tweet was excluded for being a duplicate of another tweet). Because this collection was not balanced (it has about five non-qweets for each actual qweet), they (and we) report F_1 as an evaluation measure.

3.2.2 Evaluation Measures

To compare the performance of two classifiers, we report point estimates for the evaluation measures (accuracy, F_1, and its components precision and recall), and we test differences for statistical significance using a randomization test [127, 147] constructed as follows. We consider two classifiers A and B predicting label sets L_A and L_B, respectively. Each classifier produces a point estimate (E_A and E_B, respectively) for the same measure. Under the null hypothesis, the differences in that measure between L_A and L_B are due to chance only. Consequently, we can

[5]We do not know why the other 653 tweets were missing, nor do we know whether the remaining 2,466 tweets represent an unbiased sample of the original set.

safely flip the assignment of the labels for the same document between L_A and L_B. Based on this observation, and for each document for which the labels disagree, we assign the label randomly to either of the lists. We then compute the new point estimates E'_A and E'_B. We repeat this process $n_t = 2^{20}$ times,[6] and record the number of times n_c in which the difference between the two new point estimates is greater than that between the original point estimates (i.e., $|E'_A - E'_B| > |E_A - E_B|$).[7] The significance level p is, then, at most $(n_c + 1)/(n_t + 1)$.

3.2.3 Classification Methods

In this section, we describe the preprocessing of the tweets, our neural methods, and the baselines against which we compare our results.

3.2.3.1 Tweet Preprocessing

We focus, in all of our experiments with qweet detection (including both baselines), only on the text of the tweet, deferring other attributes, such as tweet metadata and user information to future work. For both tasks, we remove the retweet term (RT) if present, and replace all of the user mentions and URLs with two special tokens, respectively. For the Arabic tweets, we additionally replace the Arabic question mark with its Latin equivalent before applying standard Arabic normalization techniques similar to the ones proposed by Darwish et al. [35]. Finally, we split

[6]This number is indicated to be large enough by [147].

[7]As we are performing a two-sided paired test, this inequality is different than the one mentioned in [147], which can be written as $E'_A - E'_B > E_A - E_B$.

the text on punctuation and special characters, adding white space before and after them. As a result, each sequence of question marks becomes a token, and hashtags are split into at least two tokens (the hash symbol "#," and one or more tokens for the corresponding word or expression).

3.2.3.2 Deep Neural Networks

Recent developments in Deep Neural Networks (DNNs) have advanced the state of the art in several text classification problems, such as factoid question answering [57, 58], sentiment [58, 69, 132] and subjectivity [69] analysis, textual entailment [17] and paraphrase detection [151]. Similarly to other classification algorithms such as support vector machines and random forests, a deep neural model has a component for predicting the labels of the classes. But DNNs distinguish themselves with an additional component for representing the tokens of the input text. In this architecture, the representation and the classification are trained jointly to satisfy the goal of the supervised task.

Although several DNN architectures have been proposed, two families became quickly prominent in text classification. Convolutional Neural Networks (CNNs) generate several transformations (a.k.a. filters), where each filter is applied on token n-grams of a fixed width in order to learn a repeating pattern [48]. Recurrent Neural Networks (RNNs) learn a single transformation for updating the representation of the entire input in a sequential order (i.e., one token at a time) [48]. Two implementations of RNNs have been predominantly used: Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU). As there is no theoretical evidence that would encourage us to favor one over the other [29], we consider both RNN variants for our classification task, in addition to CNNs.

3.2.3.2.1 Training Word Embeddings

The input to each of our neural networks is a sequence of L terms represented as an embedding vector. This vector can be pre-trained or randomly initialized. The elements of each vector are then either frozen to their initial values, or updated during the training of the network.

To train those embeddings, we use a collection of tweets from Twitter's sample stream available at the Internet Archive[8] and covering the period between October 2011 and August 2016. We rely on the "lang" field of the JSON object of the tweet provided by Twitter's API to extract 1.5B English and 296M Arabic tweets. Then we train the embeddings for each language with word2vec [99] using its default options except for the number of dimensions, which we set to $D = 300$, following previous work on sentence-level classification [58, 69, 132, 144].

3.2.3.2.2 Convolutional Neural Network

Figure 3.1 illustrates the basic structure of the convolutional network. For simplicity, we only show five dimensions instead of 300. Each sequence of N terms (e.g., $N = 3$ in this figure) goes through a convolution filter, before the activation

[8]Downloaded from `https://archive.org/details/twitterstream`. More details are provided in Section 4.2.2.1.

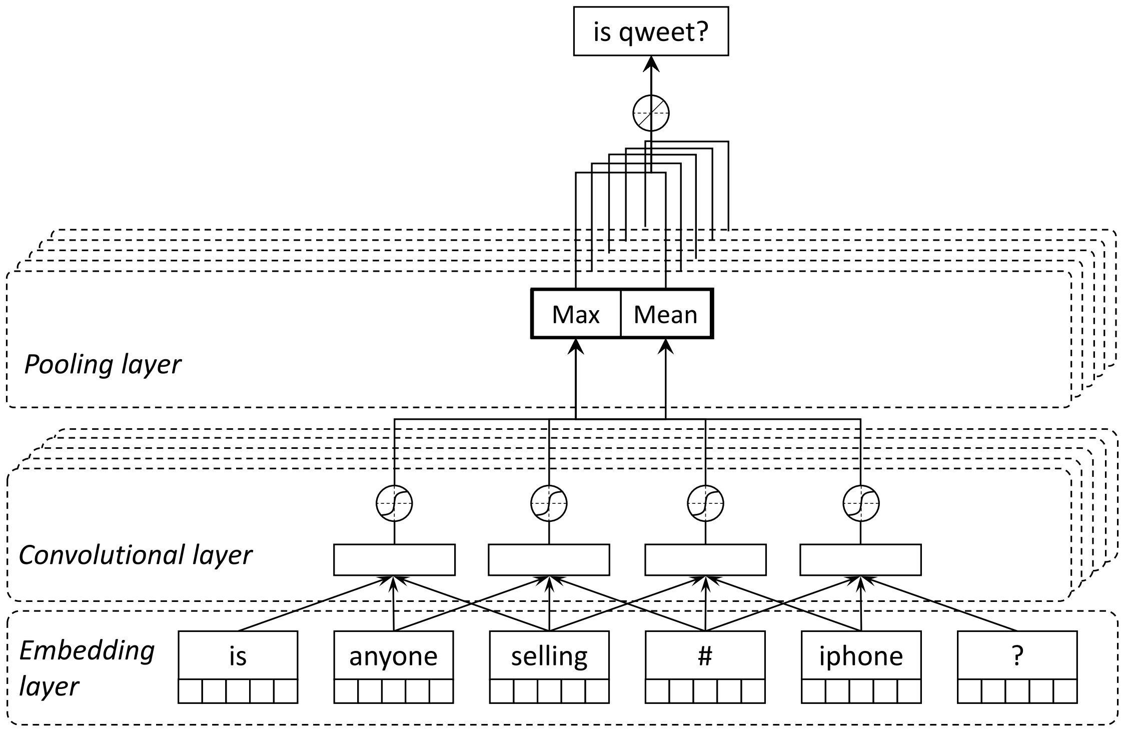


Figure 3.1: A convolutional neural network (filter length = 3).

function *tanh* is applied. The result is a sequence of $L - N + 1$ elements. We then apply max and mean pooling by selecting the maximum and mean values over all of those elements. This process was performed using one filter. We repeat it for a total of D filters.[9] The label is, then, a linear combination over a final vector of length $2D$. We optimize it with the Adam adaptive stochastic gradient descent optimizer [70], using the mean squared error as a loss function. We augment this structure by using multiple filters of different lengths (not shown), and feeding their combined output through the linear combination of the label. We refer to this architecture as $\text{CNN}_{ij...}$, where $i, j, ...$ correspond to the filter lengths.

[9]We chose the number of filters to be identical to the embedding dimension to avoid the complexity of tuning this hyperparameter.

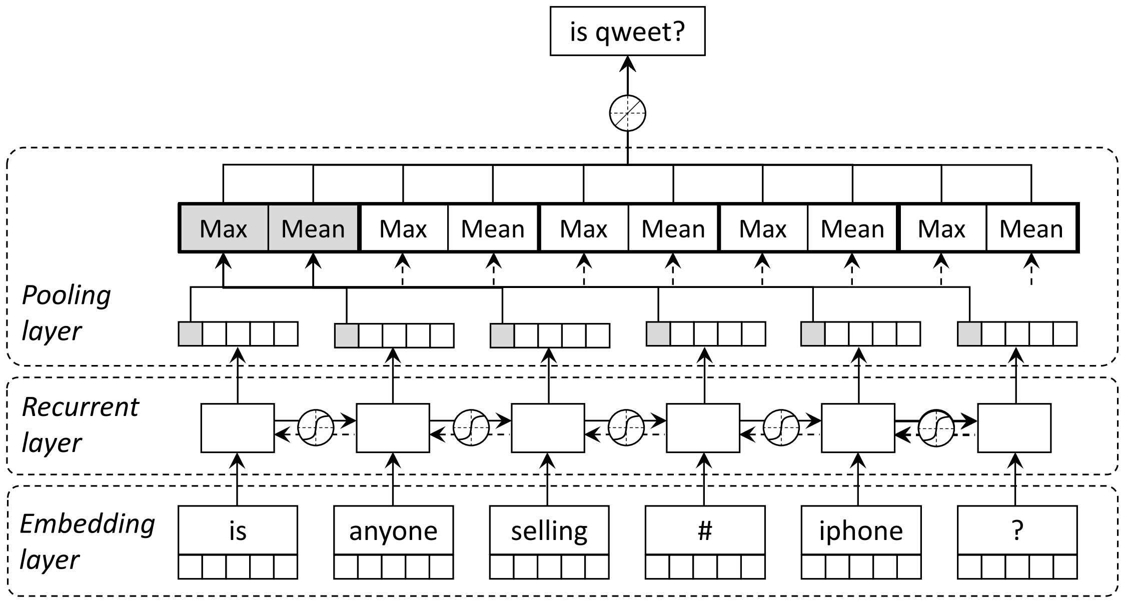


Figure 3.2: A bidirectional recurrent neural network.

3.2.3.2.3 Recurrent Neural Network

Figure 3.2 shows the basic structure of a RNN. Each term in the sequence is connected to a forward layer, followed by the activation function *tanh*. The sequence output, for which we chose that each element has the same number of dimensions D as the terms, is fed through a pooling layer, where we compute the maximum and mean, for each dimension over all elements. As with CNN, we end up with a vector of length $2D$, which is followed by a linear combination. This structure can be enriched with a backward layer, making it bidirectional. We refer to our forward-only recurrent networks as LSTM and GRU, and to the corresponding bidirectional variants as BLSTM and BGRU.

3.2.3.3 Support Vector Machines Baselines

Our baseline methods rely on a greater degree of feature engineering, some of which occurs during preprocessing. We therefore describe the baseline to which we compare for each language, including the preprocessing used by that method.

3.2.3.3.1 English Baseline

With 21% less data, we cannot replicate the results reported by Zhao and Mei [156] exactly, so we reimplemented their best system that used only lexical features to determine what accuracy can be achieved on the released set of 2,466 tweets. To do this, we first substitute a special keyword for all user mentions and then we tokenize on spaces and special characters, retaining those characters as separate tokens. We finally lower case all of the resulting tokens and apply the Krovetz stemmer [73] before generating token unigrams, bigrams and trigrams. A total of 54,408 unique n-grams result from this process.

Zhao and Mei tried two approaches to feature selection. In one approach, they selected only features that appear at least 5 times, resulting in 1,789 unique n-grams that we use as features. In their other approach, they used Bi-Normal Separation (BNS) [42] to select important features. BNS requires a hyperparameter for the number of n-grams to be selected. They found, post-hoc, that 3,119 was the best value for this hyperparameter, and that values between 3,000 and 4,000 would have been reasonable choices. We therefore used BNS to select 3,119 unique n-grams.

Following their experimental setup, we evaluated four configurations using

LIBSVM [26] with a linear kernel, and setting other parameters to their default values: (a) retaining all 54,408 unique n-grams as features, (b) using BNS alone to select 3,119 unique n-grams, (c) first removing tokens that occur only once and then using BNS to select 3,119 of the remaining 6,824 unique n-grams, and (d) retaining only the 1,789 unique n-grams that occur five or more times in the collection. We obtained the best accuracy of 0.8045 using 10-fold cross validation (CV) with configuration (a).

This result is strikingly different from the results reported by Zhao and Mei, who indicated that removing unique tokens that occur fewer than five times results in 44,121 lexical features.[10] They then apply BNS to select 3,119 of those features, and report an accuracy of 0.856 using 10-fold CV, described below. We have discussed this discrepancy with Zhao, who generously provided code that we have used as a reference for our replication. That code, however, is not the same that was used to produce the results published in their paper. Neither he nor we are able to explain the discrepancy, however. We therefore report both their reported results (on their slightly larger test collection) and those of our replication in Section 3.2.4.2.

3.2.3.3.2 Arabic Baseline

Interestingly, using different evaluation measures and a test collection with somewhat different characteristics, Hasanain et al. [54] report having found that token n-grams alone did not work well in their case for Arabic. Hence, they focused

[10]This value is close to the 48,649 (non-unique) n-gram occurrences that we observe after performing feature selection in the same way.

on augmenting n-gram features with some additional non-lexical features including structural, tweet and question-specific features. They also acquired a list of question phrases, in MSA and some Arabic dialects, to enlarge their feature set. They reported an F_1 score of 0.716 with leave-one-out CV, which is very close to the F_1 score of 0.712 we got with 10-fold CV, both of which were with the default options of SVM^{light} [60]. We observed a slightly lower performance using LIBSVM instead ($F_1 = 0.693$).

We note that Zhao and Mei also experimented with additional non-lexical features from WordNet, part-of-speech tags and tweet metadata, obtaining a 0.01 accuracy improvement. We used Hasanain et al.'s additional features when reimplementing their work because we were able to run their code, but the results we report for Zhao and Mei lack their additional features because their code was available to us only as a reference implementation, their reported gains were rather modest, and the focus of our work is on lexical features.

3.2.4 Effectiveness of Qweet Detectors

We describe the training and evaluation process for the qweet detection task before comparing the effectiveness of the classifiers.

3.2.4.1 Training and Evaluation

Following Zhao and Mei [156], we report 10-fold CV results [72]. Hasanain et al. [54] instead reported leave-one-out CV results, but the training time for our neural networks makes that evaluation design impractical for collections of this size. For each fold selected for test, we use the other nine folds for training. This produces 10% of the results that we need, so then we repeat the entire process nine more times, selecting a different test fold each time.

Using Keras,[11] with Theano as a backend,[12] we train our neural networks (but not our baselines) with an inner 9-fold CV loop. We first select one of the remaining 9 folds (10% of the total; 1/9 of the outer loop's training set) to use for validation, and we train a classifier on the remaining 8 folds. We iterate through the 8 training folds in several epochs. Within each epoch, the 8 folds, randomly shuffled, are further split into "mini-batches," each of 20 tweets. We use the ninth fold to detect when the performance (measured by accuracy or F_1, whichever is our optimization goal) on that validation fold has not improved over the best prior performance in the most recent 10 epochs, and we then select the model that had the best performance on that validation set (this will be one of those last 10, but perhaps not the very last one). Because there are 9 folds that we might have selected for validation, we then repeat this process 8 more times, selecting a different validation fold each time. The result is a committee of 9 classifiers, each with a slightly different training set. This committee then selects, by majority voting, the label to assign to each tweet in the test fold.

[11]http://keras.io

[12]http://deeplearning.net/software/theano

3.2.4.2 Results

Training neural networks can be non-deterministic for at least two reasons. First, random initialization of the weights in each layer would change with every run. Second, the associative propriety might be lost in the GPU computations, especially with single-prcision floating-point representation, as was the case in our experiments. To study this potential limitation, we fixed the random seed for all runs, and produced the results in Table 3.1 (lines 1 to 28). Using the same random seed, we produced a second batch of results (that we do not report), and then we compared the scores of the same neural network configuration between both batches of runs. We found a maximum difference of ± 0.0016 in the accuracy and F_1 score (for English and Arabic, respectively), with statistical significance at $p < 0.05$ noted in only 4 of the 56 cases. Since a Bonferroni correction [39] indicates that the overall results are not significant at p<0.05, and because none of these four cases (indicated in *italics* in that Table 3.1) involve our best system (discussed below), we analyze the results here based on our first batch of runs.

In the remainder of this results section, we start with an overview indicating the best neural configuration, and comparing it against the SVM baselines and other neural configurations. Then, we look at the individual impact of each condition of the neural networks. Namely, we compare frozen vs. updated, and random vs. trained embeddings. Then we discuss the effect of adding filters with longer lengths on the CNN architecture. Finally, we compare the variants of the RNN architectures.

Table 3.1: Systems' effectiveness on the qweet detection task. The maximum value for a given column over the classifiers we implemented is indicated in **bold**.

	Method	Embedding Init.	Embedding Weights	English Corpus Acc	English Corpus Rec	English Corpus Prec	English Corpus F_1	Arabic Corpus Acc	Arabic Corpus Rec	Arabic Corpus Prec	Arabic Corpus F_1
1	CNN_{23}	Trained	Frozen	.838	.839	.830	.834	.928	.707	**.819**	.759
2	CNN_{234}	Trained	Frozen	.835	.830	.831	.830	.927	.722	.803	.760
3	CNN_{2345}	Trained	Frozen	.832	.829	.826	.828	.926	.731	.790	.760
4	LSTM	Trained	Frozen	.845	.843	.840	.841	.929	.757	.791	.774
5	BLSTM	Trained	Frozen	**.851**	**.852**	**.844**	**.848**	**.931**	.774	.793	**.784**
6	GRU	Trained	Frozen	.843	.842	.836	.839	.929	.733	.809	.769
7	BGRU	Trained	Frozen	.835	.833	.829	.831	**.931**	.744	.813	.777
8	CNN_{23}	Trained	Updated	.834	.823	.834	.828	.907	.750	.693	.720
9	CNN_{234}	Trained	Updated	*.830*	.820	.828	.824	.907	.744	.696	*.720*
10	CNN_{2345}	Trained	Updated	.835	.824	.835	.830	.913	.735	.728	.732
11	LSTM	Trained	Updated	.819	.827	.806	.816	.905	.772	.680	.723
12	BLSTM	Trained	Updated	.825	.827	.816	.821	.906	.784	.681	.729
13	GRU	Trained	Updated	.818	.820	.809	.815	.905	.780	.679	.726
14	BGRU	Trained	Updated	.814	.814	.806	.810	.910	**.787**	.693	.737
15	CNN_{23}	Random	Frozen	.817	.801	.819	.810	.913	.679	.752	.714
16	CNN_{234}	Random	Frozen	.818	.809	.815	.812	.909	.674	.737	.704
17	CNN_{2345}	Random	Frozen	.823	.813	.821	.817	.911	.670	.751	.708
18	LSTM	Random	Frozen	.814	.830	.797	.813	.911	.728	.721	*.724*
19	BLSTM	Random	Frozen	.822	.838	.804	.821	.910	.711	.722	.716
20	GRU	Random	Frozen	.831	.830	.824	.827	.919	.694	.775	.732
21	BGRU	Random	Frozen	.818	.820	.809	.814	.921	.692	.789	.738
22	CNN_{23}	Random	Updated	*.805*	.794	.804	.798	.891	.722	.642	.680
23	CNN_{234}	Random	Updated	.816	.811	.812	.811	.898	.724	.669	.695
24	CNN_{2345}	Random	Updated	.819	.809	.817	.813	.906	.731	.698	.714
25	LSTM	Random	Updated	.807	.816	.794	.805	.890	.729	.638	.681
26	BLSTM	Random	Updated	.811	.815	.801	.808	.896	.722	.662	.690
27	GRU	Random	Updated	.798	.811	.782	.796	.891	.765	.634	.693
28	BGRU	Random	Updated	.789	.793	.778	.785	.893	.754	.642	.694
Zhao and Mei (reported in [156])				.856	-	-	-	-	-	-	
Zhao and Mei (reimplemented)				.805	.809	.793	.801	-	-	-	-
Hasanain et al. (reported in [54])				-	-	-	-	-	.644	.806	.716
Hasanain et al. (reproduced)				-	-	-	-	.903	.681	.746	.712

3.2.4.2.1 Best Configuration

The accuracy and F_1 scores in Table 3.1 suggest that the BLSTM network with trained and frozen embeddings (line 5) is the best configuration among those

Table 3.2: Effect of initializing the word embeddings randomly vs. with pre-training. Those embeddings are then either updated or kept frozen to their original weights.

Method	Shared condition on embeddings	Compared condition on embeddings					
		English			Arabic		
CNN_{23}	Frozen	Trained	$\gg$	Random	Trained	$\gg$	Random
CNN_{234}	Frozen	Trained	$>$	Random	Trained	$\gg$	Random
CNN_{2345}	Frozen	Trained	$\approx$	Random	Trained	$\gg$	Random
LSTM	Frozen	Trained	$\gg$	Random	Trained	$\gg$	Random
BLSTM	Frozen	Trained	$\gg$	Random	Trained	$\gg$	Random
GRU	Frozen	Trained	$\approx$	Random	Trained	$\gg$	Random
BGRU	Frozen	Trained	$\gg$	Random	Trained	$\gg$	Random
CNN_{23}	Updated	Trained	$\gg$	Random	Trained	$\gg$	Random
CNN_{234}	Updated	Trained	$>$	Random	Trained	$>$	Random
CNN_{2345}	Updated	Trained	$\gg$	Random	Trained	$\approx$	Random
LSTM	Updated	Trained	$\approx$	Random	Trained	$\gg$	Random
BLSTM	Updated	Trained	$>$	Random	Trained	$\gg$	Random
GRU	Updated	Trained	$\gg$	Random	Trained	$\gg$	Random
BGRU	Updated	Trained	$\gg$	Random	Trained	$\gg$	Random

we have tried. The accuracy and F_1 results on line 5 are significantly better (with $p < 0.01$) than every other neural model other than those shown on line 7 (for Arabic), and lines 4 and 6 (for both languages). Each of those statistically indistinguishable configurations involved trained frozen embeddings. Our best classifier also performs substantially and significantly ($p < 0.01$) better than our reimplmentation of the classifiers of Zhao and Mei (e.g., compare the accuracies 0.851 and 0.805) and Hasanain et al. (e.g., compare the F_1 scores 0.784 and 0.712).

3.2.4.2.2 Embedding Conditions

In Table 3.2, we set the embedding weights to be either frozen or updated during the training of the DNNs, and then we compare the effect of using the vectors trained with word2vec versus initializing the vectors randomly. We show significance test results for the differences in accuracy (for English) and F_1 (for Arabic) using

Table 3.3: Effect of freezing the embedding vectors to their original weights (which are either randomly initialized or pre-trained with word2vec), versus allowing the training of the neural network to update their values.

Method	Shared condition	Compared condition on embeddings	
	on embeddings	English	Arabic
CNN_{23}	Trained	Frozen $\approx$ Updated	Frozen $\gg$ Updated
CNN_{234}	Trained	Frozen $\approx$ Updated	Frozen $\gg$ Updated
CNN_{2345}	Trained	Frozen $\approx$ Updated	Frozen $\gg$ Updated
LSTM	Trained	Frozen $\gg$ Updated	Frozen $\gg$ Updated
BLSTM	Trained	Frozen $\gg$ Updated	Frozen $\gg$ Updated
GRU	Trained	Frozen $\gg$ Updated	Frozen $\gg$ Updated
BGRU	Trained	Frozen $\gg$ Updated	Frozen $\gg$ Updated
CNN_{23}	Random	Frozen $\approx$ Updated	Frozen $\gg$ Updated
CNN_{234}	Random	Frozen $\approx$ Updated	Frozen $\approx$ Updated
CNN_{2345}	Random	Frozen $\approx$ Updated	Frozen $\approx$ Updated
LSTM	Random	Frozen $\approx$ Updated	Frozen $\gg$ Updated
BLSTM	Random	Frozen $\approx$ Updated	Frozen $>$ Updated
GRU	Random	Frozen $\gg$ Updated	Frozen $\gg$ Updated
BGRU	Random	Frozen $\gg$ Updated	Frozen $\gg$ Updated

"$\gg$," "$>$" and "$\approx$," for $p < 0.01$, $p < 0.05$ and $p \geq 0.05$, respectively. Pre-training the embeddings appears to have a positive impact on all of the configurations for both accuracy and F_1. In fact, only 4 of the 28 configurations lack significant differences, and all of those that are significant favor pre-trained embeddings.

In Table 3.3, we set the original weights of the embeddings to be either pre-trained (with word2vec) or randomly initialized. Then we compare between the effects of freezing those weights or allowing them to be updated during the training of the neural networks. The significant differences all favor frozen embeddings, and this is clearer when the embeddings are pre-trained. Given the small size of the training set, it may be the case that the classifier overfits when updating the embedding vectors for terms seen only during training, instead of updating the

Table 3.4: Effect of CNN filter lengths on the English corpus.

Embedding		Filters		
Trained	Frozen	23	≈	234
		234	≈	2345
		23	≈	2345
Trained	Updated	23	≈	234
		234	≈	2345
		23	≈	2345

Embedding		Filters		
Random	Frozen	23	≈	234
		234	≈	2345
		23	≈	2345
Random	Updated	23	≈	234
		234	<	2345
		23	<	2345

weights of the upper layers (e.g., convolutional and recurrent layers). In other terms, while the representations of the training and test words were original learned in a similar manner (i.e., either randomly, or with word2vec), allowing the embedding weights to be updated during the training of the neural network means that only the training words might have their representations altered. If we had a substantially larger training set, we would expect the overfitting to be reduced, as there are more chances that the test words would have already been seen in the training set.

3.2.4.2.3 Convolutional Networks Filters

We show in Tables 3.4 and 3.5 the effect of adding filters with longer lengths (i.e., 4 and 5) to the convolutional network after fixing the embedding conditions. Those longer filters appear to have little effect. We observe only four cases in which there is a significant difference at $p < 0.05$. All four are improvements with random updated embeddings (i.e., lower half of the right tables). Those longer filters, perhaps, help compensate for the missing pre-training of the embeddings.

Table 3.5: Effect of CNN filter length on the Arabic corpus.

Embedding		Filters			Embedding		Filters		
Trained	Frozen	23	$\approx$	234	Random	Frozen	23	$\approx$	234
		234	$\approx$	2345			234	$\approx$	2345
		23	$\approx$	2345			23	$\approx$	2345
Trained	Updated	23	$\approx$	234	Random	Updated	23	$\approx$	234
		234	$\approx$	2345			234	$\ll$	2345
		23	$\approx$	2345			23	$\ll$	2345

Table 3.6: Effect of RNN variants on the English corpus.

Embedding		RNN			Embedding		RNN		
Trained	Frozen	LSTM	$\approx$	GRU	Random	Frozen	LSTM	$<$	GRU
		BLSTM	$\gg$	BGRU			BLSTM	$\approx$	BGRU
Trained	Updated	LSTM	$\approx$	GRU	Random	Updated	LSTM	$\approx$	GRU
		BLSTM	$\approx$	BGRU			BLSTM	$\gg$	BGRU

Table 3.7: Effect of RNN variants on the Arabic corpus.

Embedding		RNN			Embedding		RNN		
Trained	Frozen	LSTM	$\approx$	GRU	Random	Frozen	LSTM	$\approx$	GRU
		BLSTM	$\approx$	BGRU			BLSTM	$<$	BGRU
Trained	Updated	LSTM	$\approx$	GRU	Random	Updated	LSTM	$\approx$	GRU
		BLSTM	$\approx$	BGRU			BLSTM	$\approx$	BGRU

3.2.4.2.4 Comparison of Recurrent Networks

There is no clear winner between forward-only LSTM and GRU, and between BLSTM and BGRU (Tables 3.6 and 3.7). Within the sixteen comparisons, only four

Table 3.8: Effect of Bidirectional RNN on the English corpus.

Embedding		RNN			Embedding		RNN		
Trained	Frozen	LSTM	$\approx$	BLSTM	Random	Frozen	LSTM	$\approx$	BLSTM
		GRU	$\approx$	BGRU			GRU	$\gg$	BGRU
Trained	Updated	LSTM	$\approx$	BLSTM	Random	Updated	LSTM	$\approx$	BLSTM
		GRU	$\approx$	BGRU			GRU	$>$	BGRU

Table 3.9: Effect of Bidirectional RNN on the Arabic corpus.

Embedding		RNN			Embedding		RNN		
Trained	Frozen	LSTM	$\approx$	BLSTM	Random	Frozen	LSTM	$\approx$	BLSTM
		GRU	$\approx$	BGRU			GRU	$\approx$	BGRU
Trained	Updated	LSTM	$\approx$	BLSTM	Random	Updated	LSTM	$\approx$	BLSTM
		GRU	$\approx$	BGRU			GRU	$\approx$	BGRU

are statistically significant at $p < 0.05$, with GRU and BGRU winning in one case each, and BLSTM in the other two. Bidirectional RNNs do not appear to have any positive effect (Tables 3.8 and 3.9). In the only two cases where there is a statistical significance, GRU performed better than BGRU.

3.3 Detecting Answerable Qweets (Aqweets)

In a pilot study, we applied our best qweet detector (i.e., BLSTM with trained frozen embeddings, as shown in Section 3.2.4.2.1), trained on the whole corpus of 2,466 tweets released by Zhao and Mei, to the tweets that were posted in March 2013 through Twitter's sample stream, and we randomly sampled 100 tweets that were predicted to be qweets. We found that 82 of them were, indeed, questions seeking real answers, validating the high precision of our classifier. Among those, we found that 77 qweets were addressing a specific user about a personal matter (e.g., “@user Where you at?”) or required external context beyond what the text of the tweet offered (e.g., “Does anyone else have a word for today’s poetry game?”). We believe that none of these questions that convey a real information need can be answered by a stranger or a bot. That is, they are not aqweets. Hence, they should be excluded from the types of questions we would attempt to answer.

Overall, we found only five sampled qweets for which one would hope that some stranger might be able to provide a useful answer. We observed that none of these are mentioning any user, and that 71 of the 100 sampled qweets are mentioning at least one user. Therefore, we decided to exclude any tweet that has a user mention. With this process, we managed to eliminate 71/100=71% of the non-aqweets, while maintaining an aqweet recall of 5/5=100%. This recall has an approximate lower bound one-sided confidence interval [143] of 0.7831.

3.3.1 Source

We downloaded the tweets of Twitter’s sample stream from the Internet Archive for February 2016. We restricted the tweets to those that are not retweets, have a question mark and indicated in their metadata to be in English, yielding a set of itweets. We applied our best qweet detector to this set of itweets, maintaining those predicted to be qweets. To comply with Twitter’s terms of service, and to maximize the lifetime of our release, we checked each of those tweets, removing the ones that are not publicly available anymore. Following our observation from the pilot study, we removed qweets that have a user mention.

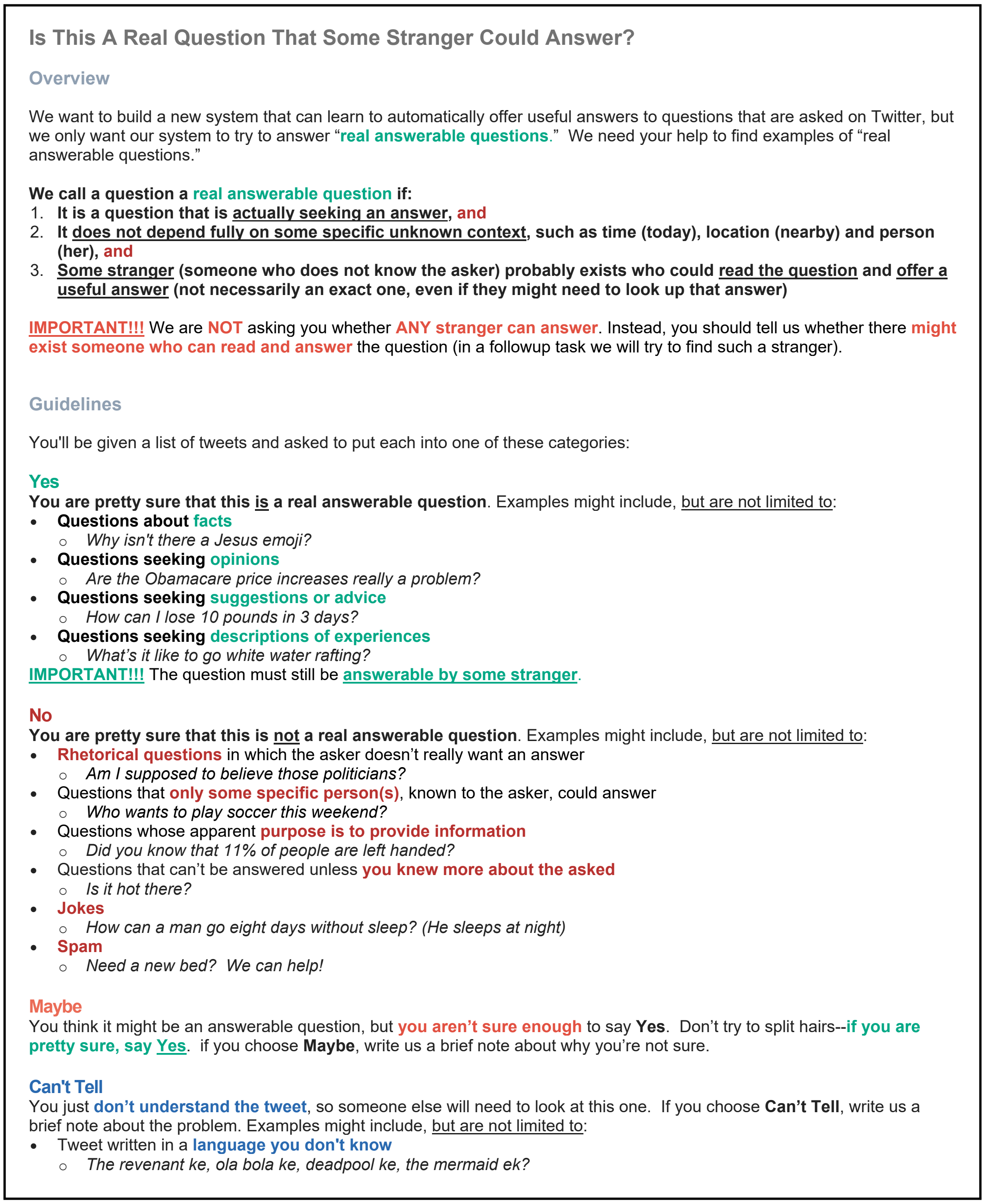
At the last stage, we obtained a list of curse words,[13] and removed all of the tweets that contain any of them, before running a deduplication on the tokenized lower-cased text of the tweets.

3.3.2 Annotations

We defined and refined the annotation guidelines through several iterations with the help of two graduate students who volunteered to assess a total of 200 tweets from March 2013. We then turned to the crowdsourcing platform CrowdFlower to gather the annotations of 5,000 predicted qweets. We titled the task "is this a real question that some stranger could answer?" and instructed the assessors to respond with "yes," "maybe," "no " or "cannot tell," with corresponding unshown scores of 1, 0.5, 0 and 0, respectively. We presented few examples with each of these options (Figure 3.3). To help the annotators to adhere to the instructions carefully, we also showed a tooltip with shortened guidelines whenever they would hover over one of the four options (Figure 3.4).

We labeled 238 tweets from a random sample of 350 tweets from March 2013, including those that were annotated by the two volunteers. Those 238 tweets were used to control the performance of the crowdsourced annotations. A candidate worker had to get all of the labels of the qualification task (5 tweets) correct, and

[13] https://gist.github.com/roxlu/1769577



Is This A Real Question That Some Stranger Could Answer?

Overview

We want to build a new system that can learn to automatically offer useful answers to questions that are asked on Twitter, but we only want our system to try to answer "**real answerable questions**." We need your help to find examples of "real answerable questions."

We call a question a real answerable question if:

1. **It is a question that is actually seeking an answer, and**
2. **It does not depend fully on some specific unknown context, such as time (today), location (nearby) and person (her), and**
3. **Some stranger (someone who does not know the asker) probably exists who could read the question and offer a useful answer (not necessarily an exact one, even if they might need to look up that answer)**

IMPORTANT!!! We are **NOT** asking you whether **ANY stranger can answer**. Instead, you should tell us whether there **might exist someone who can read and answer** the question (in a followup task we will try to find such a stranger).

Guidelines

You'll be given a list of tweets and asked to put each into one of these categories:

Yes

You are pretty sure that this is a real answerable question. Examples might include, but are not limited to:

- **Questions about facts**
  - *Why isn't there a Jesus emoji?*
- **Questions seeking opinions**
  - *Are the Obamacare price increases really a problem?*
- **Questions seeking suggestions or advice**
  - *How can I lose 10 pounds in 3 days?*
- **Questions seeking descriptions of experiences**
  - *What's it like to go white water rafting?*

IMPORTANT!!! The question must still be **answerable by some stranger**.

No

You are pretty sure that this is not a real answerable question. Examples might include, but are not limited to:

- **Rhetorical questions** in which the asker doesn't really want an answer
  - *Am I supposed to believe those politicians?*
- Questions that **only some specific person(s)**, known to the asker, could answer
  - *Who wants to play soccer this weekend?*
- Questions whose apparent **purpose is to provide information**
  - *Did you know that 11% of people are left handed?*
- Questions that can't be answered unless **you knew more about the asked**
  - *Is it hot there?*
- **Jokes**
  - *How can a man go eight days without sleep? (He sleeps at night)*
- **Spam**
  - *Need a new bed? We can help!*

Maybe

You think it might be an answerable question, but **you aren't sure enough** to say **Yes**. Don't try to split hairs--**if you are pretty sure, say Yes**. if you choose **Maybe**, write us a brief note about why you're not sure.

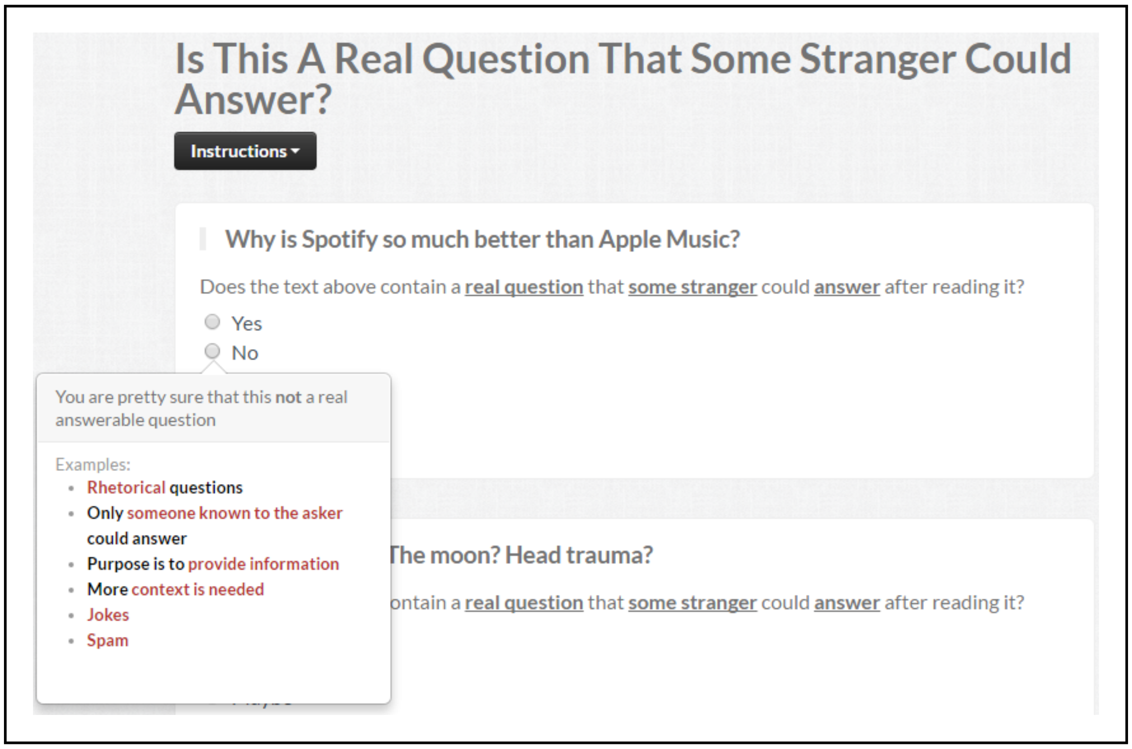
Can't Tell

You just **don't understand the tweet**, so someone else will need to look at this one. If you choose **Can't Tell**, write us a brief note about the problem. Examples might include, but are not limited to:

- Tweet written in a **language you don't know**
  - *The revenant ke, ola bola ke, deadpool ke, the mermaid ek?*

Figure 3.3: Guidelines of the aqweet detection annotation task on CrowdFlower.

maintain an average performance of 85% (which is substantially higher than the default threshold of 70%) in the subsequent pages, where we present one control and five unlabeled tweets per page. We configured the annotation job to request 7


Is This A Real Question That Some Stranger Could Answer?

Instructions ▾

Why is Spotify so much better than Apple Music?

Does the text above contain a real question that some stranger could answer after reading it?

Yes

No

You are pretty sure that this **not** a real answerable question

Examples:

- Rhetorical questions
- Only someone known to the asker could answer
- Purpose is to provide information
- More context is needed
- Jokes
- Spam

The moon? Head trauma?

ontain a real question that some stranger could answer after reading it?

Figure 3.4: Tooltip accompanying each radio box of the available annotation options.

additional assessments if any of the original 3 assessments of a tweet disagreed with the others on the four possible labels.

In total, we collected 30,447 valid assessments for the 5,000 tweets (i.e., just over 6 annotations per tweet) from 183 assessors. As our annotations are ordinal categories with missing data (i.e., not all assessors have annotated all tweets), we computed the chance-corrected agreement among the assessors using Conger's Kappa [52], which is a generalization of the simpler and more widely adopted (but, in our case, inapplicable) Cohen's Kappa measure. We found $\kappa = 0.587$, which is at the high end of moderate agreement, using magnitude guidelines provided by Landis and Koch [74].

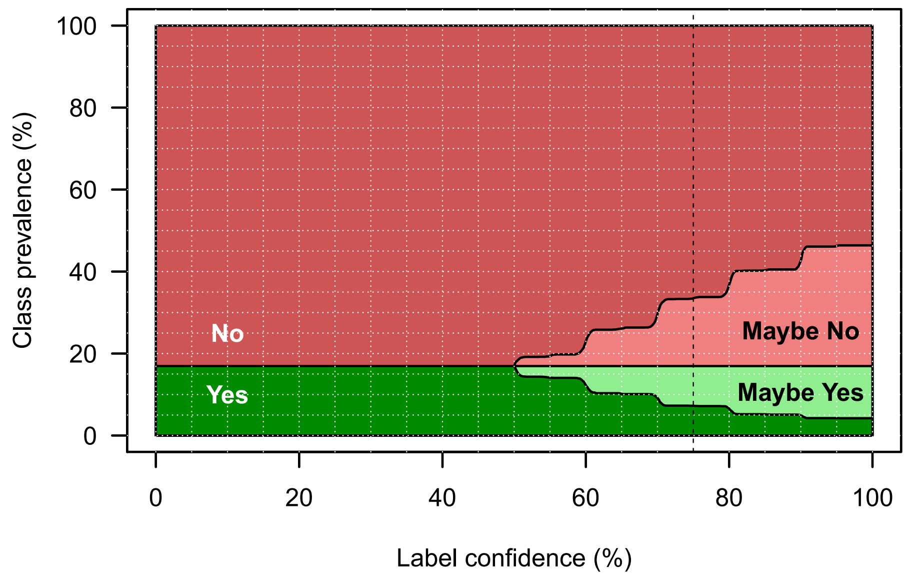


Figure 3.5: Prevalence of aqweets as a function of label confidence in a corpus of 5,000 tweets.

For each tweet, CrowdFlower computes an aggregate label with a confidence score, based on the votes of the assessors, weighted by their performance on the quality control tweets. Figure 3.5 depicts the prevalence of the aqweets as a function of the minimum confidence required. The number of positive documents range from 847 (ignoring the confidence) to 221 (when restricted to tweets with perfect agreement.) We suspect that using the tweets with a very low confidence will add noise to both training and evaluation, and that restricting ourselves to labels with very high confidence will make the evaluation too easy. Hence we decided to settle with a middle ground confidence of 75% (indicated by the vertical dashed line). At that cutoff, we have 362 positive and 3,325 negative tweets, corresponding to a prevalence of 9.82%.

3.3.3 SVM Aqweet Classifiers

While the specific problem of detecting answerable questions in microblogs, to the best of our knowledge, has not been studied before, we found some pieces of work related to the broader topic of question answerability. Uthus and Aha studied the detection of questions posted on Ubuntu's Internet Relay Chat channel that are answerable by a bot [134]. They found that an SVM classifier performs better than a k-NN classifier, and that character n-grams are more useful than word n-grams. While this work is close to ours due to the similarities between microblogs and chat messages in terms of length and language style (e.g., abbreviations, emoticons and misspellings), our problem differs in that we are not restricting our focus to factoid questions on a single domain (i.e. Ubuntu) around which a community evolved, and we are not assuming in the first place that the questions are answerable (i.e., they made the implicit assumption that all of the questions were answerable by experts). Nevertheless, that work motivated us to include character n-grams in the features that will be used by our SVM classifier.

In another domain-specific work, Yu and Sable studied the answerability of 200 questions annotated by physicians [152]. They observed a modest increase in effectiveness by using a biomedical text processing toolkit known as the Unified Medical Language System. While that system would not be useful to our case, its large database of biomedical concepts, synonyms and relations inspired our use of detected entities to enrich our feature set.

A variant of detecting question answerability was explored in the context of CQA services. Yang et al. built a classifier to predict whether a question will receive an answer in Yahoo! Answers [146]. Dror et al. extended that work to the prediction of the number of answers that a question will attract [38]. Both of these papers obtained their best performance using an SVM classifier trained on the question unigrams, in addition to other metadata features. Our universal definition of answerability is a bit different from their specific one. In particular, they consider a question for which an answer exists only outside of Y!A to be an unanswerable question. Despite this difference, and that our starting point is a tweet that is not even guaranteed to be a question, the usefulness of Y!A's metadata encourages us to explore Twitter's metadata in our task.

Following the work on detecting qweets, token n-grams seem to be a good start for the features that might be useful. Consider, for instance, words such as "she" and "her." With the absence of the name of a person, as in "Did she delete her Twitter?", it is likely that any stranger, missing some crucial context, would not be able to provide a useful answer. On the other hand "I," as in "Any awesome places in Chicago I should check out?", is a good hint that some stranger might be able to provide a useful recommendation. Hence, we use the n-grams of tokenized lower-cased terms as features, and optionally add to them their n-grams stems.

Part-of-speech (POS) tags provided a small improvement in accuracy of less than 0.01 in the qweet detection task, as reported by Zhao and Mei. We use Google's cloud natural language API[14] to extract POS tags and add them to our features.

[14] http://cloud.google.com/natural-language

Name entities can be useful as well. In “Any awesome places in Chicago I should check out?”, the mention of Chicago is a key element that makes that qweet answerable. We extract the count of each entity type detected using Google’s API. However, not all entities are equally informative. Even if the user mentions the full name of her friend, a stranger would still be missing sufficient context that would help her provide a useful answer. Thus, we also use Google’s API to extract the Wikipedia pages of detected entities. In addition, we measure the popularity of that entity as the log of the count of other Wikipedia pages linking to it.

Finally, some features can be extracted from the metadata of the tweet. We performed an ablation study, and maintained the following ones: the user id, the log of both the number of her friends and the number of lists in which she is present, the presence of geolocation information, and the indication of the device used for the publication of the tweet.

3.3.4 BLSTM Aqweet Classifiers

We saw in the qweet detection experiments that BLSTM with frozen trained embeddings is at least as good as an SVM classifier trained on hand-crafted features. We want to check whether that observation holds for the aqweet detection task as well. We adopt the same BLSTM classifier depicted in Figure 3.2, but we augment it with the non-lexical features we introduced for the SVM classifier. The corresponding architecture is presented in Figure 3.6.

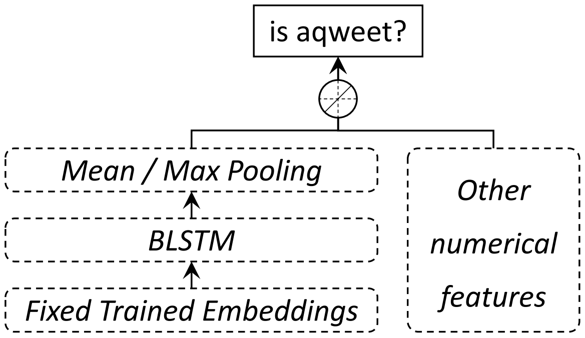


Figure 3.6: A BLSTM network with enriched features.

3.3.5 Evaluation

With a frequency of 9.82%, the F_1 score is more appropriate to use as an evaluation measure than accuracy. With this measure, a trivial acceptor (i.e., a classifier that predicts every qweet to be an aqweet) would have a score of 0.1788. Table 3.10 shows, for 10-folds CV, that both SVM and BLSTM perform much better than that simple baseline. For SVM (trained with the SVM^{perf} software, optimizing directly for F_1 [61]), term n-grams (line 1) appear to perform better than character n-grams (line 2), but with no statistical significance at $p < 0.05$. Combining them (line 4) does not seem to be helpful. In general, the incremental performance gains by adding one set of features at a time are not statistically significant at $p < 0.05$. But combined, that gain of $0.4915 - 0.4481 = 0.0434$, is statistically significant ($p \ll 0.01$). BLSTM has an F_1 score higher than all of the SVM configurations, due to the substantial difference in precision. However, a significance at $p < 0.05$ is observed only with respect to the simplest SVM model (i.e., line 1). The only feature that seems to improve BLSTM (again, with no statistical significance) is Wikipedia popularity (line 14), achieving our highest F_1 score on this task of 0.5069.

Table 3.10: Effectiveness on the aqweet detection task.

#	Model	Acc.	Rec.	Prec.	F_1
0	Trivial acceptor	0.9118	1.0000	0.0982	0.1788
1	SVM on uni/bi-grams	0.8831	0.4834	0.4177	0.4481
2	SVM on character 5/6-grams	0.8948	0.4199	0.4606	0.4393
3	SVM on stemmed uni/bi-grams	0.8693	0.4337	0.3618	0.3945
4	(1) + (2)	0.8950	0.4254	0.4625	0.4432
5	(1) + (3)	0.8942	0.4696	0.4620	0.4658
6	(5) + Tweet metadata features	0.8953	0.4917	0.4684	0.4797
7	(6) + POS counts	0.8934	0.5138	0.4615	0.4863
8	(7) + Entity-type counts	0.8931	0.5166	0.4606	0.4870
9	(8) + Wikipedia popularity	0.8940	**0.5221**	0.4644	0.4915
10	BLSTM (frozen trained embeddings)	0.9197	0.4116	0.6422	0.5017
11	(10) + Tweet metadata features	0.9222	0.4088	0.6496	0.5018
12	(10) + POS counts	0.9216	0.3978	0.6698	0.4991
13	(10) + Entity-type counts	0.9216	0.4038	0.6622	0.5017
14	(10) + Wikipedia popularity	**0.9224**	0.4061	**0.6743**	**0.5069**

The lack of significance in the difference in the F_1 score between the best SVM and best BLSTM classifiers (lines 9 and 14, respectively) hides significant and substantial differences in its two components of recall and precision. BLSTM has a superior precision of 0.6743 (compared to 0.4644), while SVM has a higher recall of 0.5221 (compared to 0.4061). Both of these are significant at $p \ll 0.01$. In the following section we attempt to improve the recall of BLSTM (and, consequently, the F_1 score), by gathering more annotations.

3.3.6 Enhancing Recall with Active Learning

After finding out that the external features contribute, at most, a limited gain to the performance of our best classifier (Section 3.3.5), we study whether

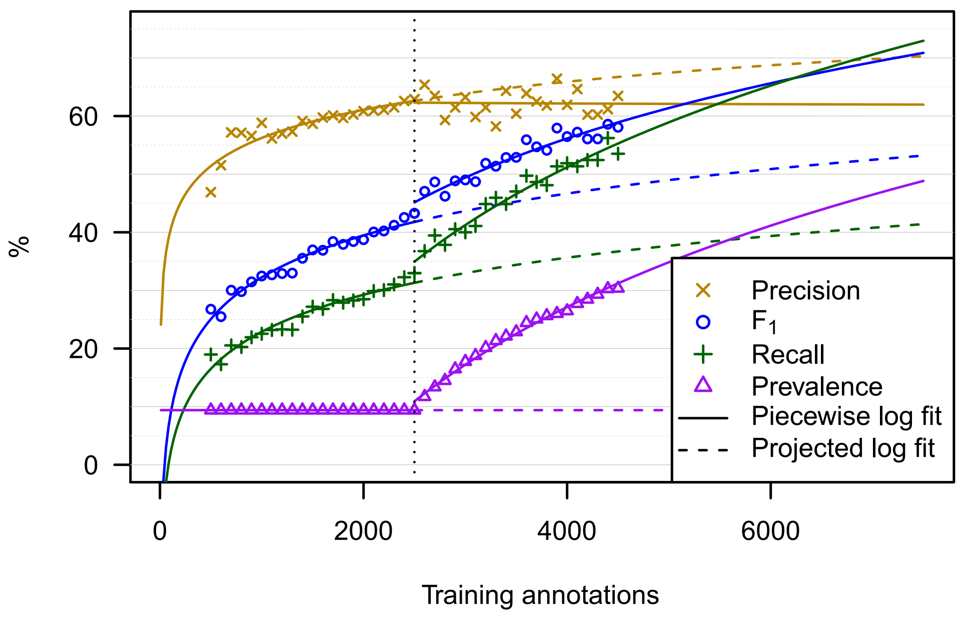


Figure 3.7: Learning curve for the BLSTM+Wikipedia model on a fixed test-set. The Training tweets are add randomly (left) or through active learning (right).

more annotations would have a substantial impact. We split our corpus into two equal halves. We reserve one for testing and use the other for training. We run a document ablation experiment on the training set, randomly removing 100 tweets, down to a training size of 500 tweets. We train our best classifier (i.e., BLSTM + Wikipedia popularity) on the training subset (after removing the tweets with a label confidence lower than 75%), and compute the corresponding recall, precision and F_1 scores. We repeat this ablation 10 times, and average the performance measures. The left portion of Figure 3.7 (i.e., less than 2,500 tweets) shows the corresponding learning curves. As recall is lower than precision, F_1 is closer to the former. All of the three performance measures increase slowly, and it appears from the projected log fits of those curves that we need to double the size of the training set to reach an F_1 score of 0.5, which is about equal to what we had obtained with 10-fold CV.

An alternative approach for gathering more useful annotations (that we restrict to training) is through active learning. We start by training a model using the whole training set. We apply that model on all of the unlabeled qweets and select a subset of 100 qweets that we send to CrowdFlower for labeling. We add this batch of 100 labeled qweets to our training set, and we repeat this process again. Among the range of possible selection methods, we consider relevance and uncertainty sampling. A small difference in performance in favor of the latter was found by Lewis and Gale in very low prevalence topics (i.e. frequency $< 1\%$) [76, 77]. However, Roegiest and Cormack recently found that the annotation behavior differs when the annotators are presented with documents selected by either strategy, compared with the other [121]. Therefore, we adopt a hybrid approach in the following way. We split the training set into 10 folds. We train 10 BLSTM models (with Wikipedia popularity), each using 9 folds for training, and 1 for validation (Section 3.2.4.1). Each model votes for the label of an unlabeled tweet. We randomly select 10 tweets out of those that get 5 positive votes (i.e., with uncertainty sampling), and 10 out of those that get 10 positive votes (i.e., with relevance sampling). Then, we randomly select 20 tweets for each bin in between (i.e., from 6 to 9 positive votes).

The right portion of Figure 3.7 (i.e., starting from 2,500 training annotations) shows the learning curves in the active learning mode. As the frequency of the positive class increases in the training set, the recall grows faster than it did in the "passive" learning mode, while the precision appears to be stable (with some variance). As a consequence, the F_1 score is increasing faster as well. After 20 iterations (corresponding to 2,000 new annotations, out of which 1,092 were at or above the label confidence threshold of 75%), the F_1 score increased from 0.4326 (recall = 0.3297, precision = 0.6289) to 0.5806 (recall = 0.5351, precision = 0.6346), attaining a maximum of 0.5859 (recall = 0.5622, precision = 0.6118) at iteration 19. Following the logarithmic fit, it appears that the learning curve has not plateaued yet, and that more annotations will still be fruitful, perhaps as long as more positive tweets are selected by the trained model for annotation.

3.4 Towards an End-to-End Pipeline

In the previous sections we introduced the individual components of a pipeline that would allow us to detect aqweets in microblogs. Next, we discuss the overall effectiveness expected for this pipeline, before pointing to some possible directions for improvement.

Figure 3.8 depicts the cascade of filters applied to a stream of tweets to detect aqweets. Previous work has stopped at the stage of qweet identification. We extended the filters to aqweet detection. Three filters (language identification and qweet and aqweet detection) are based on supervised learning. The other filters are simple and implemented through some hard coded rules, as in the case for verifying the presence of a user mention, which can be performed by checking the metadata of a tweet (which is what we did), or applying a regular expression to its content.

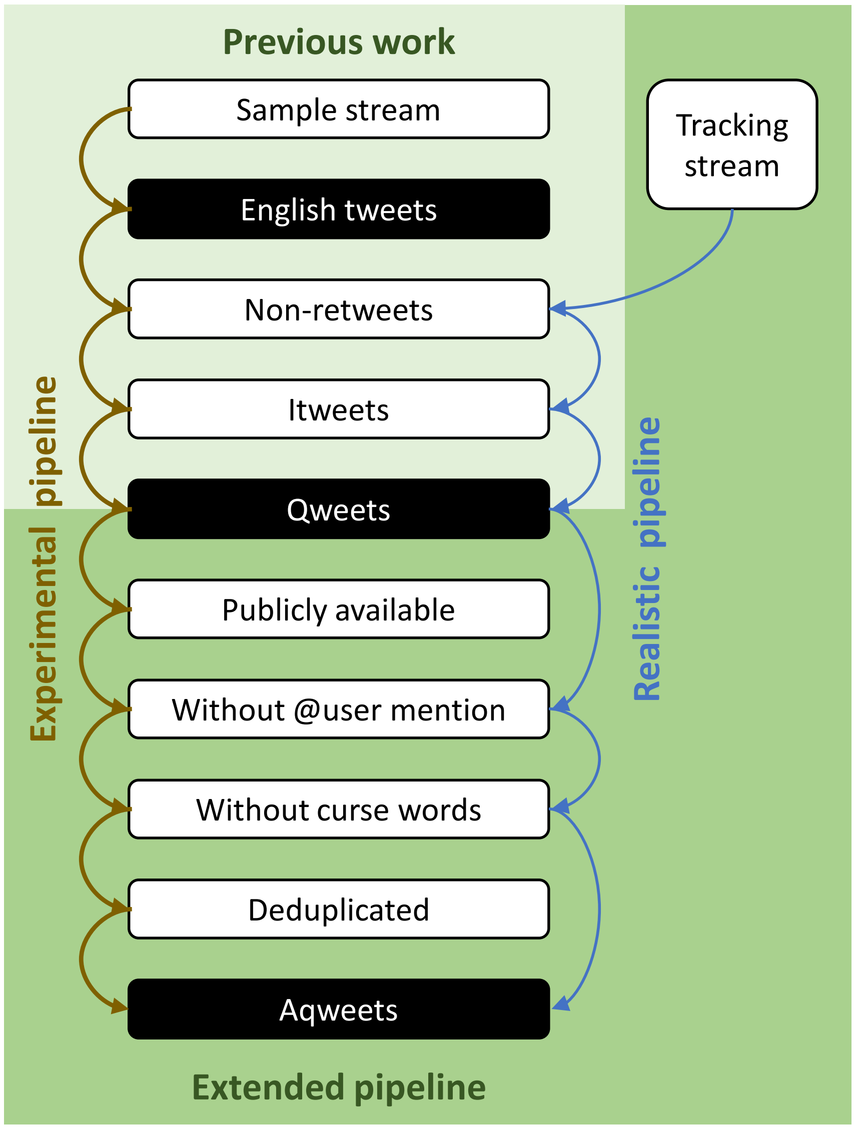


Figure 3.8: A pipeline for detecting aqweets. The black boxes indicate filters based on supervised learning, while rule-based approaches are applied for the white boxes. The top left corner corresponds to previous work. The remaining components correspond to our suggested extension. Our experiments are based on the cascade of filters shown by the the green arrows on the left side. For a real application, we suggest following the blue arrows on the right side.

3.4.1 Overall Effectiveness

Table 3.11 shows the distribution of tweets resulting from this cascade of filters.

It also shows the number of aqweets we expect to exist based on the prevalence found

Table 3.11: Effect of cascade filters on available tweets in February 2016.

Subset	Size
Sample stream	107,465,739
English tweets	36,188,015
Non-retweets	19,317,184
Itweets	1,535,437
Qweets	600,024
Publicly available	485,945
Without @user mention	104,167
Without curse words	96,068
Deduplicated	92,756
Aqweets (expected)	9,107

in Section 3.3.2. As can be observed, that number of aqweets is over five orders of magnitude smaller than the number of tweets we started with in the sample stream. One reason for that is that people post things other than just answerable questions. But a more important reason is the performance of the different stages.

We want to estimate the recall and precision of detecting aqweets in the whole pipeline. So, we start by estimating them for the individual stages (Table 3.12).[15] As some estimates are based on small collections, we indicate the point estimate and the 95% lower bound one-sided confidence interval of the recall and precision for each stage.

The sample stream is a 1% random sample of the stream of public tweets. Thus, we expect the recall and precision of this stage to be 0.01 and 1.0, respectively. Twitter automatically detects the languages of the tweets with a classifier reported

[15]For our analysis, we are not interested in retweets, tweets that are not available anymore, tweets containing curse words, and duplicates. We ignore their effect in Table 3.12, but we revisit them in Section 3.4.2.

Table 3.12: Effectiveness of individual components of the aqweet detection pipeline.

Subset	Recall		Precision	
	Lower Bound 95% Conf. Int.	Point Estimate	Lower Bound 95% Conf. Int.	Point Estimate
Sample stream	0.0100	0.0100	1.0000	1.0000
English tweets	0.8775	0.8830	0.9900	0.9900
Itweets	0.8030	0.8458	0.8236	0.8649
Qweets	0.8341	0.8518	0.8264	0.8518
No @user mention	0.7831	1.0000	1.0000	1.0000
Aqweets	0.4744	0.5351	0.5696	0.6346

to have a precision of 0.99 on English tweets.[16] However, we are not aware of any official estimate of the recall, and we might no be able to directly measure it due to a restriction on Twitter's API terms of service.[17] Hence, we approximate Twitter's official recall by the recall reported in a recent work on language identification of tweets. Lui and Baldwin indicate having achieved a macro-averaged recall of 0.883 on a dataset of 14,178 tweets across 65 languages [90]. We use this value to approximate Twitter's recall on English tweets, and the method proposed by Webber [143] to calculate the lower bound of the one-sided 95% confidence interval in recall. Finally, we include the effectiveness on the itweet, qweet, and aqweet detection tasks that we have reported in Sections 3.1, 3.2 and 3.3, respectively.

The precision of the pipeline is just the precision of the last stage (by definition, it is the number of true aqweets we detect over all of the aqweets we return). Of course, it is implicitly impacted by the earlier stages. The recall, on the other hand,

[16]https://blog.twitter.com/node/6883

[17]"Do not use, access or analyze the Twitter API to monitor or measure the availability, performance, functionality, usage statistics or results of Twitter Services." https://dev.twitter.com/overview/terms/agreement-and-policy#f-be-a-good-partner-to-twitter

Table 3.13: Comparison of itweets coverage between Twitter's sample stream and tracking stream for the day of February 16, 2017.

Subset	Sample stream	Tracking stream
Captured tweets	3,575,907	450,940
English tweets	1,187,392	450,940
Non-retweets	589,621	266,807
Itweets	47,666	228,572

is the number of true aqweets we detect over all of the aqweets that exist, subject to our restriction to non-retweets that are publicly available, have no curse words, and contain no duplicates. Assuming the distributions of aqweets in the unretrieved sequential sets is identical to those in their corresponding retrieved sets, we can estimate the overall recall by multiplying the sequence of recalls, yielding the point estimate of 0.0034.

It is possible to gather more itweets, and hence more aqweets in the subsequent stages, by tracking English tweets that have a question mark, instead of starting with the sample stream (refer to the top right box of Figure 3.8). Table 3.13 shows the number of (predicted) itweets we obtained on February 16, 2017 using both approaches. The tracking stream processed 87.39% fewer tweets than the sample stream, but captured 380% more itweets.[18] With this approach, we avoid the 0.01 recall enforced by design in the sample stream. Hence, we expect our final recall to be 0.3404.

[18]The tracking stream returns also tweets for which the question mark appears in a URL. Unless the text also contains a question mark, we exclude those from the set of itweets. This explains the discrepancy of $266,807 - 228,572 = 38,235$ between what we requested to track as itweets, and what we actually obtained.

3.4.2 Alternative Implementation Strategies

The setup discussed earlier in Section 3.4 is not the only possible implementation, as both the order and the implementation of most of the components can be modified. The elimination of retweets (as a "copy" of the original tweet) was motivated by two implicit assumptions: (1) only the original tweet would have been actually seeking an answer, or (2) the label of the original tweet would propagate to its retweets. Neither of these assumptions is necessarily true. A question that is rhetorical for its original sender might create a real information need for the retweeter. This partially explains why the annotators would sometimes disagree on the label of a tweet, and suggests that an aqweet detector does not have to be universal. That is, with sufficient training data, we could develop a personalized version of it.

Itweet detection was based, in our experiments on English tweets, only on the presence of a question mark in the content of a tweet. As we indicated in Section 3.1, Li et al. have tried a supervised approach that underperformed their rule-based method (which relied on hand-crafted rules over 5W1H words, in addition to the question mark). However, this might be a simple consequence of the small size of their corpus of 2,045 tweets, for which they had only 35 itweets with no question mark. It would be interesting to see whether a better classifier can be obtained with a larger set of tweets. We could also attempt to gather more itweets with no question mark using active learning (Section 3.3.6). In our experiments, we focused on publicly available tweets (i.e., those that did not get deleted) to comply with Twitter's terms of service, and to maximize the shelf life of our released corpus of aqweets. In a real application, for which we would provide an immediate answer after detecting the question, we could simply skip this stage. But another option is to use a classifier that predicts the deletion of aqweets. That is, if our answering component is slow to the extent that we can generate an answer only after the question is already deleted, then it is better to save some computational resources and ignore the soon-to-be-deleted aqweets. This problem of predicting the deletion of tweets in general (i.e., not aqweets) has been studied recently [8, 14, 115]. In our own work, we observed that the state-of-the-art performance is obtained by ignoring the content of the tweet, and using only the identity of the user as a feature [8]. This would be particularly worth investigating for personalized aqweet detection.

We filtered out tweets with user mentions because we observed, in a small set of 71 qweets with such mentions, that none of them is an aqweet. In fact, a lot of the discussions on Twitter that are publicly available are private in their nature. But this hard coded rule can also be replaced by a supervised classifier. Consider a user who directs her question to an expert who does not know her. She would typically phrase it with sufficient context that would make it answerable by that expert (who is a stranger). Our hard-coded rule would miss this aqweet. On the other hand, mentioning a user (by inserting the "@" symbol before her user name in a tweet) is not the only way to bring the attention of somebody. In an ongoing discussion, for instance, two or more people might use a first name (without mentioning the user name). Our hard coded rule fails to filter out such cases.

We redacted tweets that have a curse word for ethical reasons. First, we did not want to expose our annotators and future researchers to such words. Second, we preferred not to focus on such questions for the answering stage. The choice of a particular set of words was, to some extent, arbitrary. But also, some of the words might be accepted in some context and not in another. It would be more useful to use a classifier that accurately detects aqweets with appropriate content. The "appropriateness" has yet to be defined by some annotators looking at specific examples. In addition, back to the point about personalized aqweet detection, what might be appropriate for some person can be inappropriate for another one (consider for example the age, profession, health condition, religion, etc.). Thus, it would be useful to train a personalized appropriateness classifier.

Duplicate tweets share some similarities with retweets. In addition to what we discussed earlier in this section for retweets, a few other details are worth mentioning. Some near duplicate detection algorithms can be deployed instead of exact duplicate detection. Jaccard similarity is a simple technique (but not the only one) that proved itself to be useful for microblogs [93]. Duplicate (or near duplicate) detection is also useful from an efficiency standpoint. We can cache a question that becomes popular during some period of time and return an answer that would have been generated only once. The number of times a question is posted by different people is also a signal that could be used as a feature for deciding whether it is seeking a real answer, or if it is answerable.

Figure 3.4 can be augmented with other components. One such component is a classifier that predicts whether and when an aqweet will be answered by the crowd. In fact, we would be better off focusing on questions that are unlikely to get any attention, instead of those that are guaranteed to be answered (e.g., by the followers of the asker) in a short period of time. But it could also be the case that the answer we would provide enriches the diversity of the pool of answers, and is useful even with the presence of those answers.

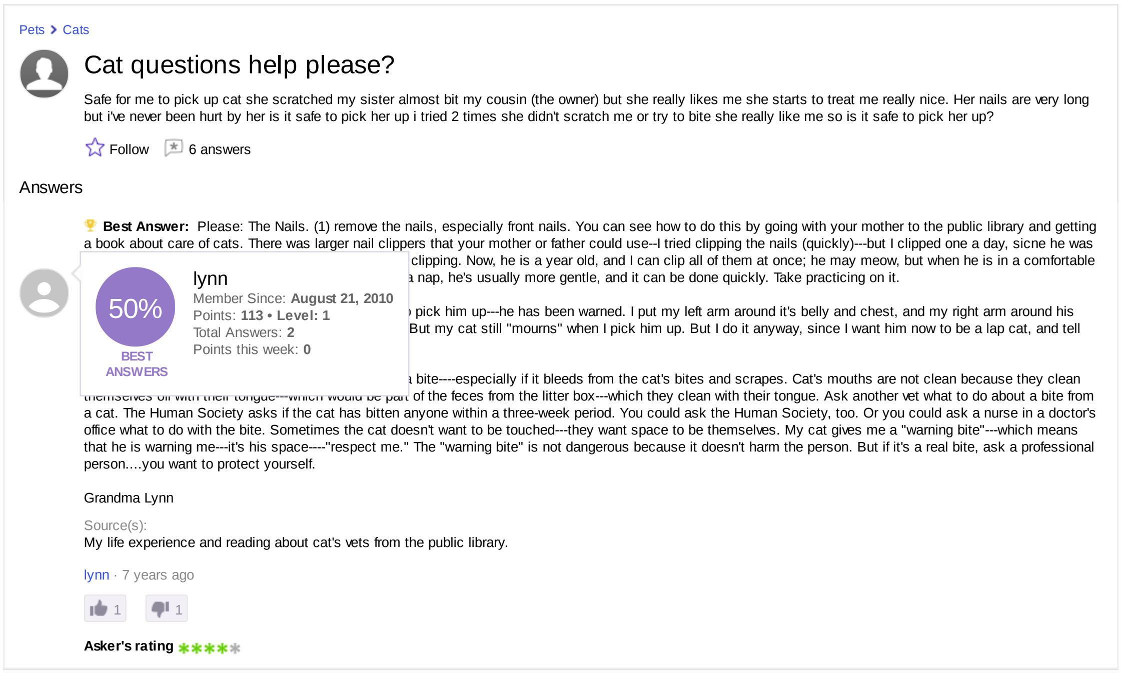
3.5 Chapter Summary

By comparing the number of questions worthy of receiving an automatic answer to the number of daily posts on Twitter, it appears that our problem is more difficult than looking for a needle in a haystack. Nevertheless, we were able to achieve a reasonable effectiveness by breaking the problem into a series of stages. We focused our experiments in this chapter on two main filters: extracting qweets (i.e., questions that seek a real information need) out of all questions, and identifying aqweets (i.e., those that can be answerable by a stranger, and thus we would hope that a bot might be able to provide some useful answer). A BLSTM neural architecture works fairly well on both stages. We use it to automatically detect questions asked on Twitter, before attempting to answer them (in Chapter 4).

Chapter 4: Finding Useful Answers[1]

A large fraction of questions asked on social media attract poor answers or remain unanswered. Even those that receive good answers might do so after a long period, or could suffer from a redundancy of the answers. In this chapter, we study how to harvest past and future user-generated content to automatically and quickly return answers to the asker. We address some of the issues stated above in the context of two TREC evaluation campaigns (Section 4.1): the Live Question Answering (LiveQA) track, which evaluates system that can provide answers in less than one minute to complex questions posted in Yahoo! Answers, and the Real-Time Summarization (RTS) track, which focuses on detecting novel relevant tweets as soon as they are published. We also introduce a new task of answering questions asked on microblogs using content from community question answering websites (Section 4.3). Our methods (Section 4.4 and 4.5) are based on two large collections of user-generated content that we crawled with a special attention to exhaustiveness (Section 4.2).

[1]Some parts of this chapter were extracted from publications by Bagdouri and Oard [6, 9, 10].


Pets > Cats

Cat questions help please?

Safe for me to pick up cat she scratched my sister almost bit my cousin (the owner) but she really likes me she starts to treat me really nice. Her nails are very long but i've never been hurt by her is it safe to pick her up i tried 2 times she didn't scratch me or try to bite she really like me so is it safe to pick her up?

Follow 6 answers

Answers

Best Answer: Please: The Nails. (1) remove the nails, especially front nails. You can see how to do this by going with your mother to the public library and getting a book about care of cats. There was larger nail clippers that your mother or father could use--I tried clipping the nails (quickly)---but I clipped one a day, sicne he was clipping. Now, he is a year old, and I can clip all of them at once; he may meow, but when he is in a comfortable a nap, he's usually more gentle, and it can be done quickly. Take practicing on it.

lynn
Member Since: **August 21, 2010**
Points: **113 • Level: 1**
Total Answers: **2**
Points this week: **0**
50% BEST ANSWERS

pick him up---he has been warned. I put my left arm around it's belly and chest, and my right arm around his But my cat still "mourns" when I pick him up. But I do it anyway, since I want him now to be a lap cat, and tell

a bite----especially if it bleeds from the cat's bites and scrapes. Cat's mouths are not clean because they clean of the feces from the litter box---which they clean with their tongue. Ask another vet what to do about a bite from a cat. The Human Society asks if the cat has bitten anyone within a three-week period. You could ask the Human Society, too. Or you could ask a nurse in a doctor's office what to do with the bite. Sometimes the cat doesn't want to be touched---they want space to be themselves. My cat gives me a "warning bite"---which means that he is warning me---it's his space----"respect me." The "warning bite" is not dangerous because it doesn't harm the person. But if it's a real bite, ask a professional person....you want to protect yourself.

Grandma Lynn

Source(s):
My life experience and reading about cat's vets from the public library.

lynn · 7 years ago

1 1

Asker's rating ★★★★☆

Figure 4.1: A question posted on the *Pets* category, and its selected best answer.[2]

4.1 TREC Evaluation Campaigns

Our participation in two tracks of the NIST TREC evaluation campaign has allowed us to study several aspects related to searching, selecting, returning and presenting answers that address some information needs.

4.1.1 TREC Live Question Answering Track

TREC LiveQA is a new track that loosely follows the earlier TREC QA track, but with several substantial design changes [1, 2]. In this new track, the questions come in real time from real users, as posted on Yahoo! Answers. A question, such as the one illustrated in Figure 4.1, has a title, an optional description (or body), and belongs to a hierarchy of categories. This results in more natural and diverse topics than was the case with earlier QA tasks in which the questions were 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). This constraint models a user who is eager to receive a fast answer, especially when future user-generated answers will be posted late. 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. In addition to being a challenge, this lack of a shared data collection is an opportunity to study how the contents from different platforms might be useful to answer different questions.

The first edition of this track focused on eight categories, namely *Arts & Humanities*, *Beauty & Style*, *Health*, *Home & Garden*, *Pets*, *Sports*, *Travel* and *Computers & Internet*. The last category was dropped in the second year due to difficulties in annotations. The systems were required to return only one answer per question of no more than 1,000 characters, which then was judged on a 4-level scale (0 = bad, 1 = fair, 2 = good, and 3 = excellent). These restrictions of one answer and 1,000 characters are not particularly interesting to our work. In fact, some questions require detailed answers that exceed this limit. Also, we would be better off having several candidate answers judged. However, we think that the advantages of this track setup exceed those limitations. Hence, we use our participation in the two editions of this track as an opportunity to study the performance of three systems that provide answers from two sources: a crawl of old

[2] https://answers.yahoo.com/question/?qid=20100821220150AAkU07g

questions and answers from Yahoo! Answers, and a large collection of tweets. We try to mitigate the limitations of the track by summarizing long answers, concatenating short answers, and introducing a new approach for evaluating answers that have not been annotated (Chapter 5).

4.1.2 TREC Real-Time Summarization Track

Performance of information retrieval systems is typically assumed to be non-negative, which is reflected in measures such as recall, precision, MAP and nDCG. This makes sense in an interactive search task, where the user has initiated the communication with a search engine (e.g., through a query) and is waiting for a response. The user, in this case, has already dedicated some time for her search session, and whatever she obtains as results will not decrease her prior information (although it might indirectly create new information needs). As a consequence, developers of retrieval systems are mainly concerned about returning a good ranking of documents. But there are also cases in which the user would not expect a response at any specific time, but she would want to receive a good answer whenever one becomes available. In this case, the system would interrupt the user with that answer. This interruption would have a cost (e.g., distraction of the user from whatever else she is doing), and should be allowed only when the reward is sufficiently high (e.g., the answer is highly useful). Another example is if we would provide an unsolicited answer to a tweet we have detected to be a question with a real information need. If that answer is very good, then perhaps we shouldn't worry much. But if we pro-

Table 4.1: Topic MB229 of the TREC 2015 Microblog track [84], that was also used in the TREC 2016 Real-Time Summarization track [85].

Title	bus service to NYC
Description	Find information on bus service to NYC.
Narrative	The user needs to travel to New York City (NYC) and is considering using the bus to get there. He is looking for other passengers' opinions of the various bus lines regarding aspects such as the quality of service, economic value, driver safety, cleanliness, reliability, and safety of pickup/drop off locations.

vide a bad answer, or we naively answer a rhetorical question, then the user might consider our reply to be a spam. Hence, there is a need to model some penalty for providing bad answers, and to train the retrieval system to return a candidate answer only if the reward is expected to exceed the risk.

An experimental setup that allows us to study a variant of this problem is the "push scenario" (a.k.a. Scenario A) of the TREC 2016 Real-Time Summarization (RTS) track [85], which originally started in the TREC 2015 Microblog track [84]. In this task, a user has an interest in some broad topic (see Table 4.1 for an example), and wants to stay up to date in that topic using a stream of microblog posts. To do so, a system should monitor that stream in real time and 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. The track ran for 10 days from August 2 to 11, 2016 (UTC), and was based on Twitter's sample stream.

Clearly, the broad topic is not necessarily a question, and the topical relevance of a tweet does not necessarily translate into an answer usefulness. However, those approximations allow us to study the interesting problem of making a binary decision for whether to return a document, when a penalty might be imposed if that document turns out to be not useful.

4.2 Collections of Answers

While typical evaluation of question answering (and document retrieval) systems relies an a corpus of documents in which systems are restricted to search, more realistic setups relax this restriction, adding the complexity of gathering some content that might potentially be useful. The TREC LiveQA track, for instance, requires the participants to assemble any online or offline resources on which their systems will base their answers [1, 2]. Similarly, for answerable questions posted on microblogging services such as Twitter, we need to decide where to search for answers. This section describes two major sources of potential answers: a substantial crawl of Yahoo! Answers, and a large corpus of tweets.

4.2.1 A Crawl of Yahoo! Answers

A study by Shtok et al. has shown that about 25% of the question titles 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 [125]. 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 questions), then we might be able to find useful answers to new questions by searching in old questions and answers. For this reason, 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, illustrated in the upper section of Figure 4.2:

1. Crawl the pages of all categories of the main website, in addition to its 22 localized versions[3] 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 who asked or answered 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 obtain the identifiers of the questions they asked or responded to, which we add to the set Q, and to acquire 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 260M questions and 1.4B answers from 49M 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

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

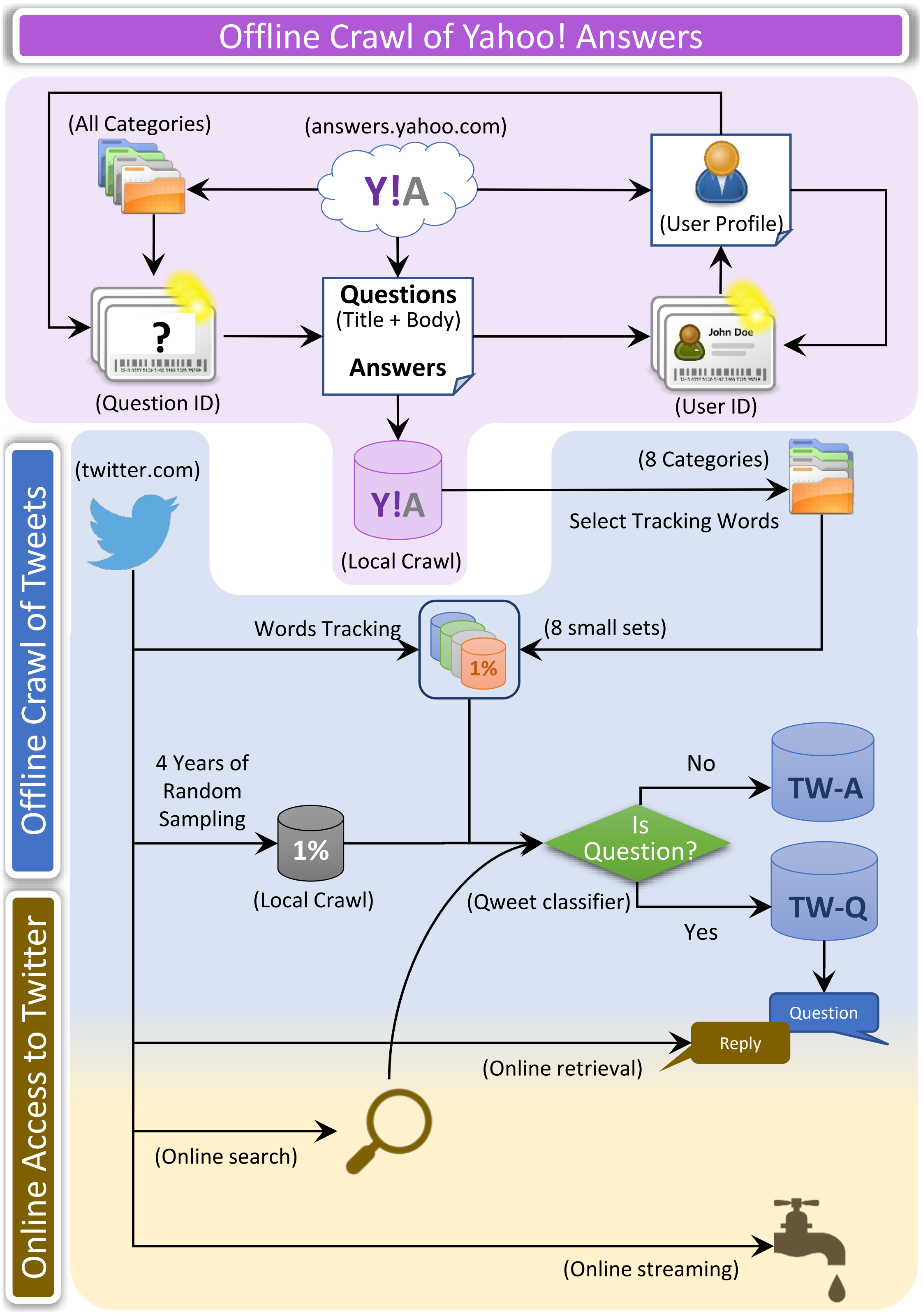


Figure 4.2: Collections of tweets and Y!A questions and answers.

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.

We limit our focus to the 123M questions and 673M answers downloaded from the main Yahoo! Answers website,[4] which are expected to be in English. 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 ignore the questions and answers from those localized websites because we do not expect many useful matches to be found there.

4.2.2 Collections of Tweets

Twitter, like several other popular social networking services, 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 some questions. However, we do not have access to all of the tweets that have ever been posted. In this section, we describe some ways for increasing our chances of gathering tweets that will be useful for the expected questions. They are illustrated in the bottom section of Figure 4.2.

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

4.2.2.1 A Large Corpus of Random Tweets

A random sample of 1% of Twitter's public stream is accessible through Twitter's API as JSON objects.[5] While this is a small portion of all tweets published at any given time, an accumulation over a long period can help to collect a large number of them. The Internet Archive Team has been collecting these tweets for several years.[6] Because of some technical difficulties, tweets sent on some days are missing from this collection. Hence, we only have tweets from 1,452 days within the period September 27, 2011 to March 31, 2016. We add to these a collection of tweets that we obtained using the streaming API between April 1, 2016 and May 22, 2016. We extract over 1.6B English tweets (using the "lang" field of the JSON object). We denote this collection *SampleStream*.

4.2.2.2 A Small Corpus of Selected Tweets

If we know in advance the distribution of words of a particular set of questions, we can collect tweets that contain specifically those words, instead of a random sample of tweets. In fact, the Twitter API allows us to track[7] a set of up to 400 keywords.[8] 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 TREC LiveQA categories (Section 4.1.1), we think that selecting some

[5]https://dev.twitter.com/streaming/reference/get/statuses/sample

[6]https://archive.org/details/twitterstream

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

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

keywords that represent its *core vocabulary* and then tracking tweets that contain 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 sample stream.

We construct these eight core vocabularies following Fung et al. [44]. 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 a set of documents G. We then scale $DF(w_G)$ to a value between 0 and 1 as:

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

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 and $\overline{C}$ are two instances of G, such that C contains all of the documents that belong to the category i, and $\overline{C}$ contains all of the remaining documents.

This value gives higher credit to words that are more frequent within the category i than within all of the other categories. In other terms, 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

Table 4.2: Words selected from Yahoo! Answers to be tracked in Twitter.

Arts & Humanities	Beauty & Style	Computers & Internet	Health
book	hair	computer	doctor
books	wear	windows	weight
poem	color	laptop	diet
novel	skin	download	exercise
writing	makeup	pc	body
author	dress	software	fat
story	style	install	muscle
twilight		click	pain
characters		files	eating
authors		program	calories
(13 more)		(11 more)	(5 more)

Home & Garden	Pets	Sports	Travel
plant	dog	team	travel
paint	dogs	football	city
wood	vet	players	trip
walls	pet	win	hotel
plants	cat	teams	airport
garden	cats	fan	flight
wall	breed	player	hotels
furniture	puppy	wwe	cities
depot	pets		tourist
soil	animals		places
(65 more)	(13 more)		(14 more)

1% of all the posted tweets. The final set of keywords for these eight categories is presented in Table 4.2. We tracked these eight[9] sets of keywords using eight Twitter accounts for three weeks (in TREC 2015 LiveQA) and 55 days (in TREC 2016 LiveQA). We denote each of the eight corresponding collections by $\mathit{TrackedWords}_i$, while i corresponds to the name of the category.

[9]The organizer of the TREC 2016 LiveQA track removed the category *Computers & Internet*. Thus, we did not track the corresponding words in that year.

4.2.2.3 Questions and Answers in Twitter

Oftentimes, tweets that are retrieved following a search operation are themselves questions. Such tweets should not be returned to the asker as answers to her 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 are detected to be questions seeking an answer using our reimplementation of the qweet identifier of Zhao and Mei (Section 3.2.3.3). Their replies, extracted online should a qweet be retrieved, would potentially contain some useful answers.

- $Corpus_{i,a}$ is the subset of tweets that are not detected (by that same classifier) to be qweets. Such tweets are themselves potential answers.

We denote by **TW-Q** and **TW-A**, respectively, the unions of corpora $Corpus_{i,q}$ and $Corpus_{i,a}$.

4.2.2.4 Recent Tweets

Hoping to cover questions about current events, we add a third source of tweets. To do so, we start by extracting term bi- and tri-grams from the question. Then, 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. To avoid exceeding the 60 seconds answering time allowed in the TREC LiveQA track, we stop when the total time spent in communicating with Twitter reaches 40 seconds.

4.2.2.5 Future Tweets

The TREC RTS track has the special particularity of monitoring a stream of tweets to detect novel relevant tweets. In other words, we aim to retrieve "future" tweets with respect to the topic of interest. Per the task's guidelines, that stream of tweets is accessible through Twitter's sample API. Since there is no offline collection in which we search, we need to approximate document frequencies (typically used to weight the terms of a query) from an external source. We use the statistics of the *SampleStream* corpus to this end.

4.3 Answering Aqweets

In this section, we investigate the ability to answer microblog questions by searching in Yahoo! Answers. Our goal is to return a "thread" (e.g., an old question with its answers) that the asker might find useful. We start with the set of 362 aqweets (i.e., answerable question tweets) we had collected in Section 3.3.2, which we split into 177 training and 185 test aqweets. Then, we study different search strategies using our crawl of Yahoo! Answers and considering some transformations that can be applied on the questions. Next, we study how to select among the various available configurations in a learning-to-rank (L2R) framework. After that, we examine how to combine the L2R answers and the replies that the aqweets might have received over time. We compare the effectiveness of our techniques against two baselines, one is based on the aqweet replies, and the other returns the answers using the internal search engine of Yahoo! Answers.

4.3.1 Where to Search

Our techniques for searching for answers use our crawl of Yahoo! Answers (Section 4.2.1), which we limit to the questions posted on or before December 31, 2015 (i.e., at least one month prior to the posting time of the aqweets). We also exclude crawled questions and answers that contain any term from the same list of curse words we had used for filtering aqweets (Section 3.3.1).

A thread in Yahoo! Answers has several fields in which we can search. One possibility we consider is the concatenation of the title and body of the question, as well as all of its answers. This approximates a simple search for a Web page by a search engine. Alternatively, we can index each field separately. This allows us to study the importance of every field independently from the others, and examine different combinations.

There are two possibilities for indexing the fields of a thread. In the first, we index each field of the question, in addition to the concatenation of all of its answers. We call this indexing setup question-per-document, and refer to it as QpD. In the second possibility, the indexed document contains the fields of the question, and one answer at a time. That is, we index as many documents for a given thread as the number of its answers. We call this indexing setup answer-per-document, and refer to it as ApD.

The fields we index are the title (T) and body (B) of the question, their concatenation (C) and the answer (or concatenation of answers for QdP) (A). We experiment with various combinations of these four indexed fields. Finally, each configuration returns the top-1 thread. We break ties by selecting the most recent thread (this is often needed when we restrict our search to the title field, because different questions may share the same title).

4.3.2 Scoring Function

We adopt the BM25 ranking scheme for our retrieval of answers [63]. There are few variants of this ranking function. We use the one implemented in Lucene 6.4.2 as BM25Similarity. For a term t from a question q, let the term frequency $TF(t,d,f)$ be the number of times the term t appears in the indexed field f of a thread d. The document frequency $DF(t,f)$ is the number of threads in which the term t appears at least once within the field f. The inverse document frequency $IDF(t,f)$ is defined, provided the number of indexed threads of the field N_f, as:

$$IDF(t,f) = log\left(1 + \frac{N_f - DF(t,f) + 0.5}{DF(t,f) + 0.5}\right).$$

BM25 scores the thread d with respect to the term t and the field f as:

$$bm25(d,t,f) = IDF(t,f) \times \frac{TF(t,d,f) \times (k_1 + 1)}{TF(t,d,f) + k_1 \times \left(1 - b + b \times \frac{len(d,f)}{len(f)}\right)},$$

where the length $len(d,f)$ is the count of the occurrences of all terms in the field

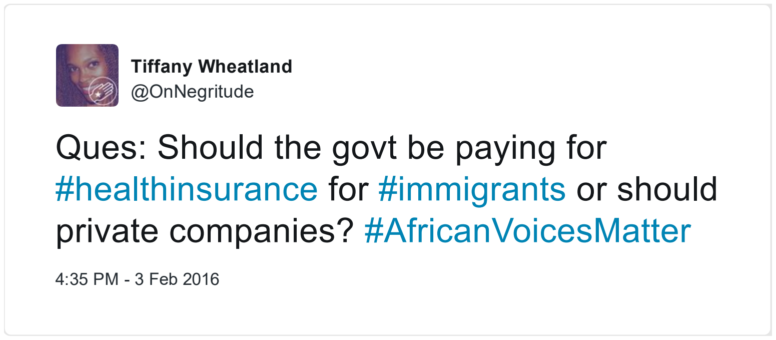
f of thread d, $\overline{len(f)}$ is the average length over all threads of that field, k_1 is a free parameter that controls the saturation of the term frequency, and b is a free parameter that controls the document length normalization. Their default values in Lucene are set to 1.2 and 0.75, respectively.

Finally, for a set of fields F, we compute the BM25 score of the thread d with respect to the question q as:

$$BM25(d, q, F) = \sum_{f \in F} \sum_{t \in q} bm25(d, t, f).$$

4.3.3 Question Rewriting

Tweets experience some characteristics that are unusual in other platforms, such as misspellings, abbreviations and multi-words hashtags (e.g., Figure 4.3). We address each of these challenges with a dedicated rewriting method that generates one or a list of candidate replacements. Candidate generation involves using a character n-gram language model built by indexing character 1-4-grams from the vocabulary of the Twitter *SampleStream* corpus (Section 4.2.2.1). Candidate selection involves using a word n-gram language model created as a positional Lucene index built from the same corpus, where we add special START and END tokens at the boundaries of the tweets, and tokenize them using Lucene's standard analyzer (without stemming) with an empty stop list.


Tiffany Wheatland
@OnNegritude

Ques: Should the govt be paying for #healthinsurance for #immigrants or should private companies? #AfricanVoicesMatter

4:35 PM - 3 Feb 2016

Figure 4.3: Example of a question requiring several transformations.[10]

4.3.3.1 Hashtag Normalization

Twitter users often use hashtags to highlight a special notion. A hashtag cannot contain a space. Hence, users would concatenate the terms of a multi-word expression into a single hashtag. Some times they would do so using the *CamelCase* convention, as is the example for #AfricanVoicesMatter in Figure 4.3. In other instances, they would not bother alternating between upper and lower cases, as with #healthinsurance in the same tweet. We expect hashtag segmentation to improve retrieval performance, but we should be careful about what to segment. For instance, #immigrants should be kept as a single word in that tweet.

On the other hand, not all hashtags are equally useful. Some of them substitute for regular words (or expressions) within the content of the question, as is the case for both #healthinsurance and #immigrants. Other hashtags occur after the end of the question (i.e., after the question mark), providing some context, but without being part of the question itself. #AfricanVoicesMatter is an example.

[10] https://twitter.com/OnNegritude/status/695042981242486784

We normalize the hashtags following two steps. First, we remove the hashtags that appear after the last question mark. Then, we perform a segmentation over the remaining hashtags. Our approach for segmenting a candidate hashtag is based on three stages. In the first stage, we remove the # symbol, and use Google's cloud natural language API to check if that term is detected within the question, as an entity of any type besides OTHER. This stage aims to avoid segmenting single-word proper names. In the second stage, we generate one or more candidate segmentations. If a hashtag follows the CamelCase convention (detected with a regular expression),[11] we extract the corresponding segmentation. Otherwise, we use the vocabulary of our Twitter index to extract all possible segmentations, with a maximum of 3 words per candidate segmentation[12] (a limit is helpful to restrict the number of candidates, especially for long hashtags, and in some words it could be set dynamically as a function of the hashtag length). Some of the candidate segmentations would be invalid (e.g., segmenting #iPhone into "i phone"). Hence, in a third stage, we filter out all segmentations that appear (as a sequence) in Twitter's positional index less frequently than the hashtag (without the # symbol). If no segmentation passes this filter, we maintain the hashtag (but remove the # symbol). Otherwise, we replace it with the segmentation that has the highest frequency (breaking ties arbitrarily).

[11]We use the regex below suggested in http://stackoverflow.com/questions/2559759:
`(?<=[A-Z])(?=[A-Z][a-z])|(?<=[Â-Z])(?=[A-Z])|(?<=[A-Za-z])(?=[Â-Za-z])`

[12]We use the WordBreakSpellChecker.suggestWordBreaks() method of Lucene 6.4.2.

4.3.3.2 Spelling Correction

Twitter is mostly accessed from mobile devices.[13] The tiny keyboards on those devices increase the chance of misspellings in the posted tweets. Consider for example the question "Why did the great awaking happen?"[14] We have little hope for finding an answer, or even a similar old question, 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 (as in the example above), causing us to miss the relevant threads even if we would go deep in the retrieved list. Another potential impact might appear when a high frequency word (e.g., a stop word) is misspelled as a rare word with high IDF. An example of such a case is "should igo to school tomorrow?",[15] where *igo* leads to the undesirable retrieval of threads about intergovernmental organizations.

We perform spelling correction in three stages. As with hashtag segmentation, we exclude from this process terms that are detected to be entities. In the second stage, we generate a list of up to 1,000 closest words with Levenstein's distance, using the character n-grams index.[16] In the third stage, we maintain only alternatives for which both their document frequencies and the document frequencies of their n-gram context, are greater than those of the original word. The n-gram context is constructed by appending the left and right words or special START and END

[13] `http://venturebeat.com/?p=2014007`

[14] `https://twitter.com/HistoricEmily/status/699268868351664129`

[15] `https://twitter.com/xChipmunk_/status/317080343700197376`

[16] We use the SpellChecker.suggestSimilar() method of Lucene 6.4.2

markers, until a non-stop word is encountered, or no additional words are available. If one or more alternatives pass this filter, we return the alternative with the highest document frequency as a synonym to the original word.

Once a spelling correction is found, we consider it to be a synonym of the original word, and compute the BM25 score, for a particular field, after summing their term frequencies in that field, and approximating their combined document frequency by the maximum of their individual document frequencies (which is, by construction, the document frequency of the alternative word).

4.3.3.3 Synonyms

The informal language of tweets encourages the adoption of some writing conventions that are less frequent in other places. For example, *you* and *conversations* would be valid (and perhaps even better) synonyms to *u* and *convos*, for the question "Should u read your kids convos on the Internet?"[17] The process of finding synonyms is also divided into three stages. The first (filtering entities) and the third (checking the frequency of the synonym and its context) are identical to what we did for spelling correction (Section 4.3.3.2). For the second stage (suggesting a candidate synonym), we use a word2vec [99] model trained on our SampleStream corpus (Section 4.2.2.1) to suggest the nearest word to the original one, but only if the cosine similarity of their vectors exceeds a threshold of 0.5.

[17]`https://twitter.com/DeadDinero/status/318060250542518272`

4.3.4 Term Statistics

The importance of a term is indicated, in the BM25 scoring function, by its IDF (Section 4.3.2). As a result, the same term might have different IDF values in different indices. For the question “What am I gunna do with this dog for the night?”,[18] we observed that *night* has a high IDF in Yahoo! Answers compared to *dog*, although it appears that the latter is more important than the former for this particular question. Using the document frequencies from a Twitter index (built based on the *SampleStream*), we had the opposite observation of having a higher IDF value for *dog*. Hence, some words seem to suffer from a “cost of fame.” That is, they are so important, that so many questions are asked about them in Yahoo! Answers (e.g., there is an entire subcategory for questions about dogs), diminishing their IDF. We suggest using the IDF statistic from our index of tweets, which will have the same value for all of the fields.

4.3.5 Question/Question Similarity

The same question might be phrased in different ways, even within the same platform. Hence, it would be useful to detect if an aqweet is, semantically, a duplicate of 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.[19] We use 90% of those pairs to train the neural network depicted in Figure 4.4, and

[18] `https://twitter.com/TheFizzyBubbly/status/317901609373937665`

[19] `http://qim.ec.quoracdn.net/quora_duplicate_questions.tsv`

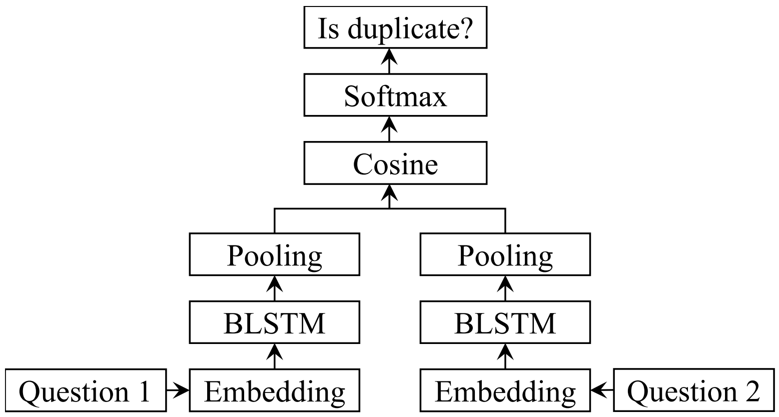


Figure 4.4: An architecture for detecting duplicate questions.

the remaining validation subset to stop training when the accuracy does not improve over the best prior performance in the previous 10 epochs. We return the model that has the best accuracy (0.855) on that validation set, after optimizing it with the Adam adaptive stochastic gradient descent optimizer [70], using the mean squared error as a loss function, as implemented in Keras, backed by Theano.

4.3.6 Learning to Rank Threads

The approaches we have introduced so far for searching in our crawl of Yahoo! Answers aim to find the configurations that would work best, on average. However, it is also possible to use features of the questions and the answers to select the thread to be returned among the ones that were retrieved by different configurations. We operationalize this selection in a learning-to-rank (L2R) framework. We start with all of the single-field (i.e., T, B, C, A or P) retrieval models, including those where one or all of the question rewriting techniques were applied, and those for which alternate term statistics were used. For each aqweet, we retrieve the union of the top-1 threads of these retrieval models. Then we score all of threads in this union using all of the retrieval models. That is, if two retrieval models RM_i and RM_j return two different threads d_i and d_j, then we compute a total of 2×2 BM25 scores. At the end of this stage, we represent each thread as a vector of BM25 scores, where each element corresponds to one retrieval model.

Next, for each thread, we add the following features:

- The Jaccard coefficient and Quora-based similarity between the aqweet and the title of the thread.
- The minimum, maximum and mean of the similarity between the aqweet and each of the answers of the thread, using both Jaccard coefficient and Quora-based similarity.
- The number of answers in the thread.

We standardize each feature (including the BM25 scores) by applying the z-score transformation (learned from all threads of the training aqweets, and applied to all threads of the training and test aqweets). Finally, we train a L2R model based on the threads of the training aqweets using the SVM^{rank} software [62]. We use that model to select the final top-1 thread for each aqweet in the training and test sets.

4.3.7 Combining Twitter Replies and Yahoo! Answers

People might react to the questions they see on Twitter by replying to them. Some of those replies might contain useful answers. Hence, the asker could already have some level of satisfaction when there is no intervention on our part. In general, the longer the asker would wait, the better the answers she is expected to receive become. For this reason, we looked at the replies returned within 1 hour, 1 day and 1 year. If the asker has received an answer outside of the replies we have collected (e.g., a private message, a deleted reply, a tweet that is not a reply), then we cannot observe that. Thus, the asker satisfaction estimated based on the observed replies is, in reality, a lower bound of the actual user satisfaction. For simplicity, however, we will restrict ourselves to the replies we observed.

When a tweet receives no reply at all, it is obvious we should always attempt to return the best thread we can obtain.[20] When there are some replies to a given aqweet, then the decision for returning a thread from Y!A depends on how we model the satisfaction of the user if she is presented with answers from the two sources (i.e., Twitter and Yahoo! Answers). One model we consider can be expressed as:

$$satisfaction(Twitter, Y!A) = max\{satisfaction(Twitter), satisfaction(Y!A)\}.$$

With this model, we should always return the best thread we have. A more conservative (and interesting) model is expressed as:

$$satisfaction(Twitter, Y!A) = min\{satisfaction(Twitter), satisfaction(Y!A)\}.$$

In this case, we need to (automatically) decide whether our best retrieved thread should be returned or not. In the remainder of this section, we restrict ourselves to

[20]We implicitly assume that there is no penalty for returning an unsolicited bad answer.

this latter user-satisfaction model.

We suggest two methods for combining the answers. In the first method, we always prefer Twitter's replies when they exist. That is, we return a thread only for aqweets that have not received any reply. In the second method, we train a classifier (on the training aqweets) that learns to make this decision. The model is trained only on the questions for which the scores of Twitter replies and our top-1 thread differ.[21] For an aqweet with its replies and a candidate thread, we use the following six features, based on the Jaccard similarity and the similarity predicted by the model trained on the Quora corpus:

- The similarity between the aqweet and the title of the thread.
- The maximum similarity between the aqweet and its replies.
- The maximum similarity between the aqweet and the answers of the thread.

4.3.8 Baselines

We consider two reasonable baselines for answering aqweets: the replies that the question receives on Twitter, and a search using Y!A's internal search engine.

4.3.8.1 Twitter Replies

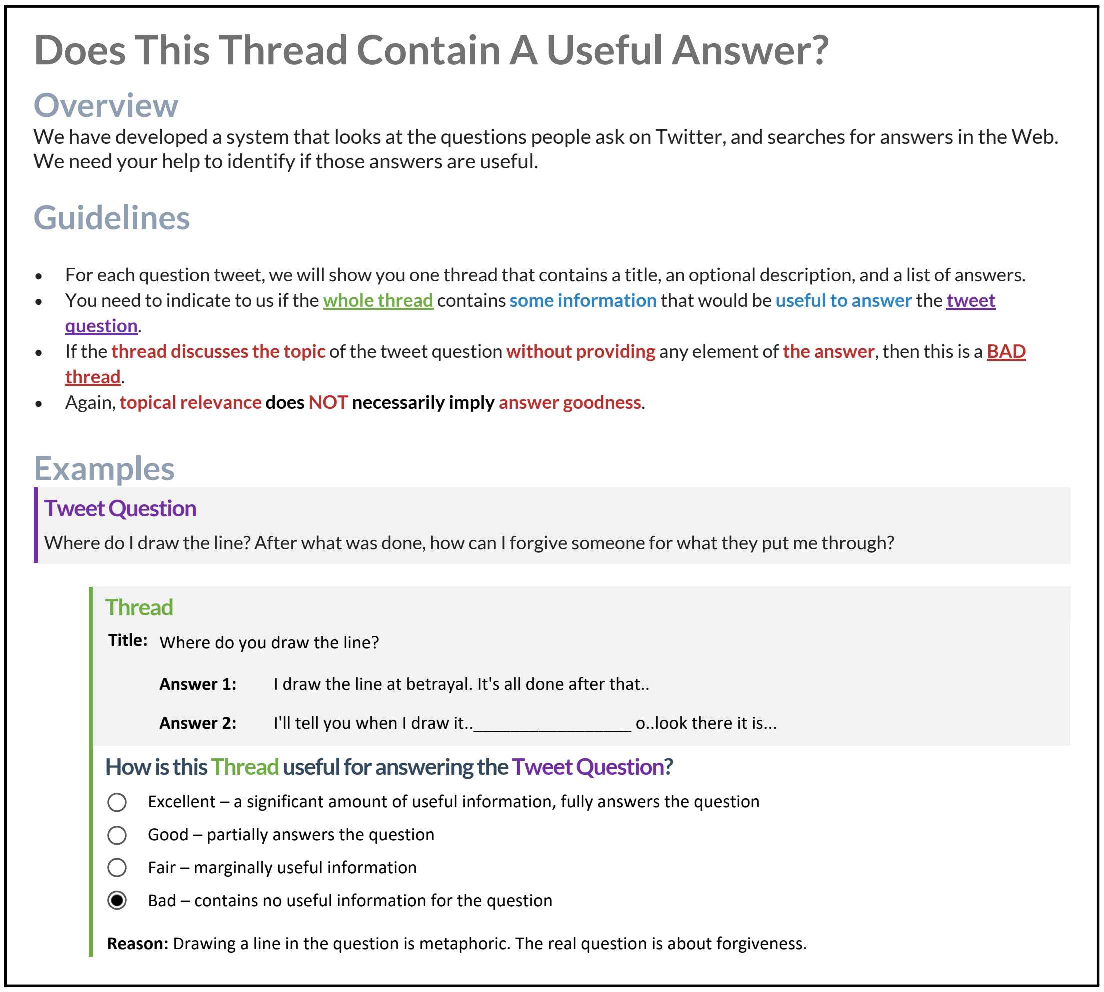
The first baseline we look at is the set of replies that the question receives over time (Section 4.3.7). We crawled the web page of each aqweet, extracting the content and publication time of every available reply. We found that 89 aqweets had

[21]We use SVM^{light} with its default options for training and prediction.

at least one reply within one hour of their publication time. This number increases as we allow longer waiting periods, reaching 102 aqweets with one or more reply within one day, and 110 aqweets with one or more reply within one year.

4.3.8.2 Yahoo! Answers' Internal Search

The second baseline we consider is the result of a search using Yahoo! Answers' internal search engine. The ranking of the results in that engine might not be tuned for aqweets. In fact, for 187 out of 362 aqweets we issued to the engine as queries (with no modification), no thread was retrieved at all. In such cases, we removed one term at a time from the question, alternating between its start and end, before issuing it again to the search engine. We consider four configurations for this online Yahoo! Answers search. In the first one, we restrict the retrieved results to the ones we have also included in our indexed crawl. This enables a direct comparison of the respective ranking functions. In the second configuration, we restrict the retrieved results to the same period covered in our crawl (i.e., until December 31, 2015). This allows us to validate the exhaustiveness of our crawl. In the third configuration, we include all of the threads that were posted anytime prior to the posting date of the aqweet. This is particularly useful for questions about recent events. Finally, in the fourth configuration, we relax all of the restrictions to include even the threads that were posted after the aqweet's posting time.


Does This Thread Contain A Useful Answer?

Overview

We have developed a system that looks at the questions people ask on Twitter, and searches for answers in the Web. We need your help to identify if those answers are useful.

Guidelines

- For each question tweet, we will show you one thread that contains a title, an optional description, and a list of answers.
- You need to indicate to us if the whole thread contains some information that would be useful to answer the tweet question.
- If the thread discusses the topic of the tweet question without providing any element of the answer, then this is a BAD thread.
- Again, topical relevance **does NOT necessarily imply** answer goodness.

Examples

Tweet Question

Where do I draw the line? After what was done, how can I forgive someone for what they put me through?

Thread

Title: Where do you draw the line?

Answer 1: I draw the line at betrayal. It's all done after that..

Answer 2: I'll tell you when I draw it..________________ o..look there it is...

How is this Thread useful for answering the Tweet Question?

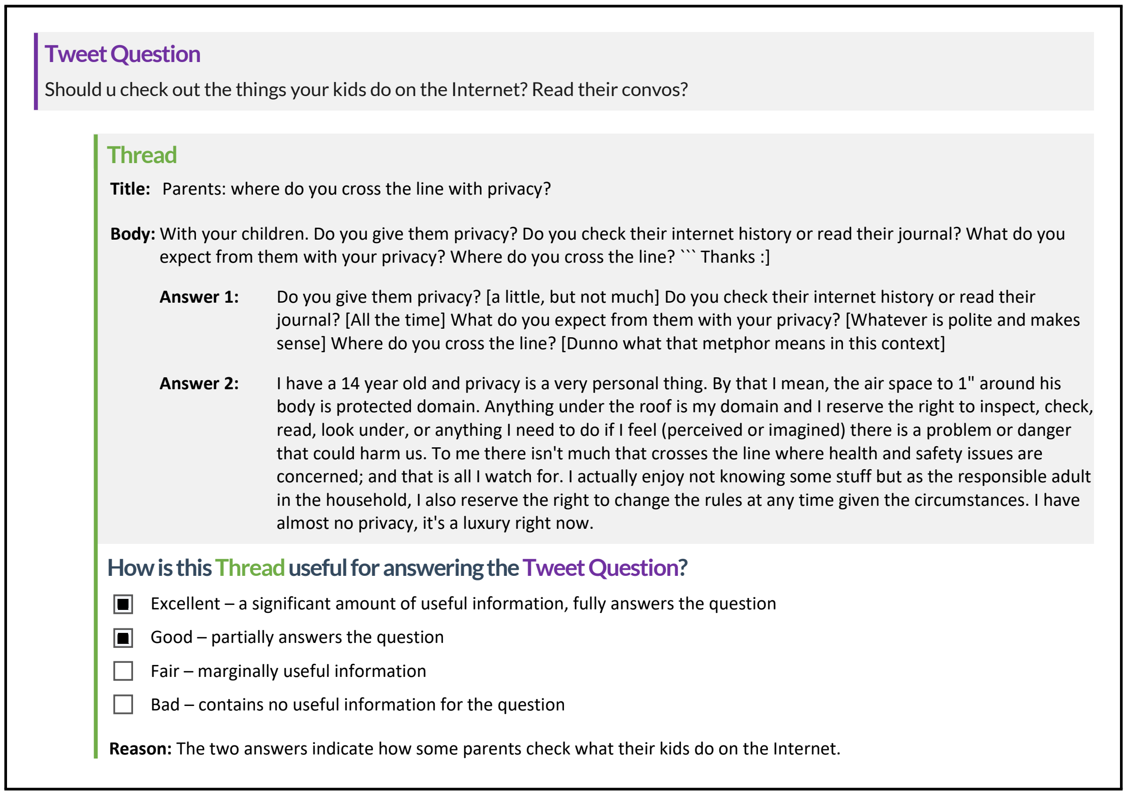
- ○ Excellent – a significant amount of useful information, fully answers the question
- ○ Good – partially answers the question
- ○ Fair – marginally useful information
- ● Bad – contains no useful information for the question

Reason: Drawing a line in the question is metaphoric. The real question is about forgiveness.

Figure 4.5: Guidelines for the aqweet answering annotation task on CrowdFlower.

4.3.9 Annotations

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 aqweet. In practice, it is difficult for the annotators to assess the relevance of a long thread. Thus, we decided to present the thread using its question title and body, and a small number of answers. One answer is selected as the best-answer, if one exists, or as the one with the highest difference between the thumbs-up and thumbs-down, breaking the ties by the score of the relevance model. Another answer is selected


Tweet Question

Should u check out the things your kids do on the Internet? Read their convos?

Thread

Title: Parents: where do you cross the line with privacy?

Body: With your children. Do you give them privacy? Do you check their internet history or read their journal? What do you expect from them with your privacy? Where do you cross the line? ``` Thanks :]

Answer 1: Do you give them privacy? [a little, but not much] Do you check their internet history or read their journal? [All the time] What do you expect from them with your privacy? [Whatever is polite and makes sense] Where do you cross the line? [Dunno what that metphor means in this context]

Answer 2: I have a 14 year old and privacy is a very personal thing. By that I mean, the air space to 1" around his body is protected domain. Anything under the roof is my domain and I reserve the right to inspect, check, read, look under, or anything I need to do if I feel (perceived or imagined) there is a problem or danger that could harm us. To me there isn't much that crosses the line where health and safety issues are concerned; and that is all I watch for. I actually enjoy not knowing some stuff but as the responsible adult in the household, I also reserve the right to change the rules at any time given the circumstances. I have almost no privacy, it's a luxury right now.

How is this Thread useful for answering the Tweet Question?

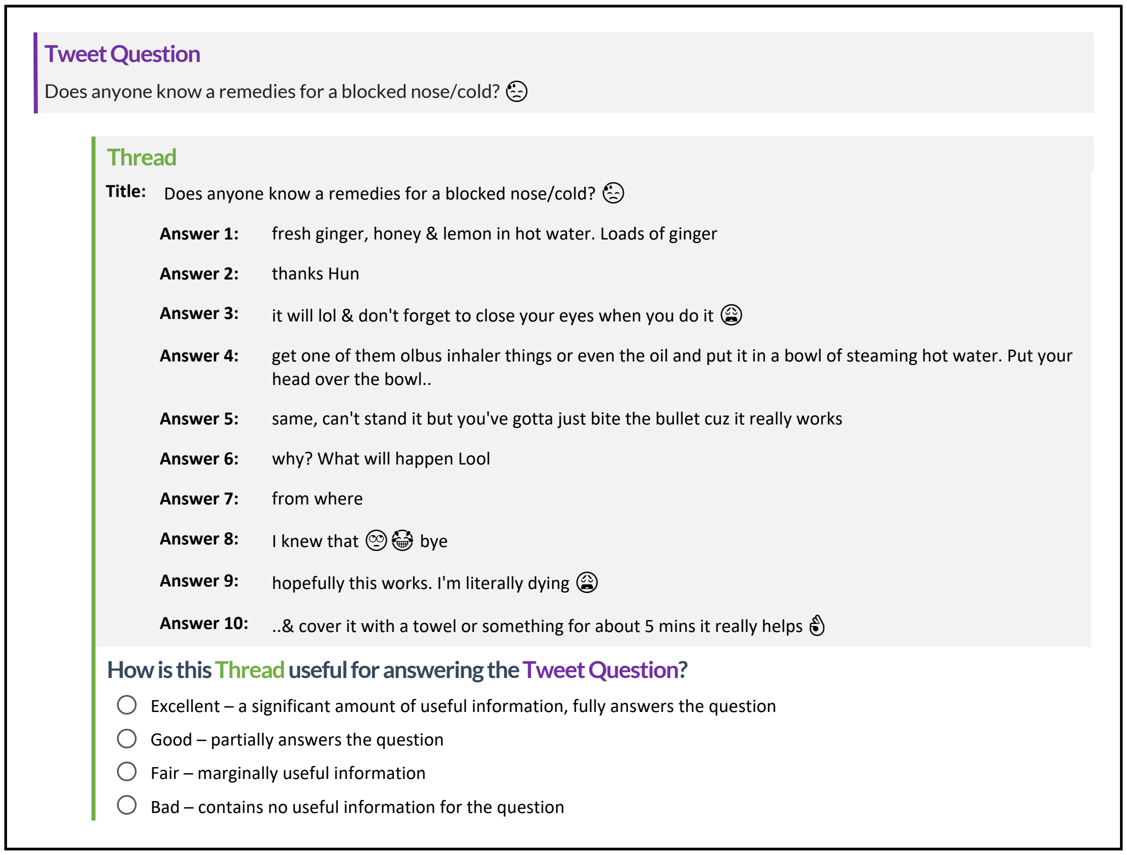
- [✓] Excellent – a significant amount of useful information, fully answers the question
- [✓] Good – partially answers the question
- [ ] Fair – marginally useful information
- [ ] Bad – contains no useful information for the question

Reason: The two answers indicate how some parents check what their kids do on the Internet.

Figure 4.6: A test pair where either of two checked boxes would be a good label.

by the relevance model used to find that particular thread. In the rare case when multiple relevance models retrieve the same thread but disagree on the scores of the answers, then we include the union of these top-1 answers in the pool of answers presented in the thread.

We instantiated the annotation task on CrowdFlower and requested the annotators to assess the relevance of a thread to an aqweet on a 4-level scale (Figure 4.5), following the setup of the two editions of the TREC LiveQA tracks [1, 2]. We provided a reference example for each of the four options. We prepared 200 aqweet/thread pairs, which we used for a pre-qualification stage, where the annotators were required to pass a test with a score of 6/6. Then, as they proceeded within the annotation task (5 unlabeled pairs + 1 test pair per page), they had to maintain


Tweet Question

Does anyone know a remedies for a blocked nose/cold? 😓

Thread

Title: Does anyone know a remedies for a blocked nose/cold? 😓

Answer 1: fresh ginger, honey & lemon in hot water. Loads of ginger

Answer 2: thanks Hun

Answer 3: it will lol & don't forget to close your eyes when you do it 😩

Answer 4: get one of them olbus inhaler things or even the oil and put it in a bowl of steaming hot water. Put your head over the bowl..

Answer 5: same, can't stand it but you've gotta just bite the bullet cuz it really works

Answer 6: why? What will happen Lool

Answer 7: from where

Answer 8: I knew that 😳😂 bye

Answer 9: hopefully this works. I'm literally dying 😩

Answer 10: ..& cover it with a towel or something for about 5 mins it really helps 👌

How is this Thread useful for answering the Tweet Question?

- ○ Excellent – a significant amount of useful information, fully answers the question
- ○ Good – partially answers the question
- ○ Fair – marginally useful information
- ○ Bad – contains no useful information for the question

Figure 4.7: A thread where candidate answers are replies to an aqweet.

an accuracy of at least 85%. Due to the subjectivity of the relevance assessment, we sometimes would accept more than one label for a test pair (the annotators would still make a single choice). This is illustrated in Figure 4.6.

The replies to the aqweet are represented as a thread as well. We replicate the content of the tweet in the title of the thread and hide the description field. We present each reply as an independent answer in a chronological order (Figure 4.7).

The final label of the aqweet/thread pair is aggregated as a weighted average score over three annotators, where the weights correspond to the performance of these annotators over the test questions. We use those labels to report the results of Section 4.3.10.

4.3.10 Results

We experimented with a total of 65 configurations, in addition to the baselines and the L2R models. Table 4.3 shows the average top-1 accuracy (over the scale [0-3]) for a subset of configurations. First, we note the low average scores of the Twitter answers (lines 1 to 3), due, to a large extent, to the absence of any replies for most of the questions. Our own simple search, which considers the thread to be a single concatenated page (line 8), performs significantly better than all of the Twitter "systems" on the training set ($p < 0.05$ with a 2-sided paired t-test). Its score is also higher on the test set, but we lose significance as we allow more time for the replies ($p < 0.05$ for 1 hour, $p < 0.1$ for 1 day, and no significance for 1 year).

There is an incremental increase in the scores of the Yahoo! Answers online search systems (e.g., from 0.75 to 0.81 on the test set) as we relax the restrictions on the threads to be retrieved. None of the differences, however, is statistically significant. Our own simple search (i.e. with QpD-P) is comparable to Yahoo! Answers' internal search systems, as none of the differences is statistically significant.

Looking at the combinations of fields, we observe that the single best field is the title (line 9). In both sets, it is significantly better than the body and the answer fields (lines 10 and 15, with $p < 0.01$), as well as the page field (line 8, with $p < 0.05$). But its outperformance over the concatenation of the body and answer fields (line 12) is significant only on the training set ($p < 0.05$).

Comparing the question-per-document indexing setup and the answer-per-document setup (compare lines 13, 14 and 15 to 16, 17 and 18), the former ap-

Table 4.3: Effectiveness of aqweet answering configurations over the scale [0-3].

#	Configuration	Parameter	Average score Training	Test
1	Twitter future replies	1 hour	0.40	0.63
2	Twitter future replies	1 day	0.51	0.68
3	Twitter future replies	1 year	0.53	0.76
4	Yahoo! Answers online search	Crawled subset	0.71	0.75
5	Yahoo! Answers online search	Crawled period	0.75	0.77
6	Yahoo! Answers online search	Recent QAs	0.74	0.80
7	Yahoo! Answers online search	Future QAs	0.77	0.81
8	BM25	QpD-P	0.74	0.86
9	BM25	QpD-T	1.19	1.15
10	BM25	QpD-B	0.74	0.74
11	BM25	QpD-TB	1.02	1.01
12	BM25	QpD-C	1.03	1.05
13	BM25	QpD-TA	1.11	1.16
14	BM25	QpD-TBA	1.01	1.05
15	BM25	QpD-A	0.65	0.64
16	BM25	ApD-TA	0.97	1.07
17	BM25	ApD-TBA	0.87	1.00
18	BM25	ApD-A	0.46	0.50
19	BM25 + Hashtag Normalization	QpD-T	1.21	1.14
20	BM25 + Hashtag Normalization	QpD-TA	1.14	1.16
21	BM25 + Spelling Correction	QpD-T	1.19	1.16
22	BM25 + Spelling Correction	QpD-TA	1.12	1.16
23	BM25 + Syonyms	QpD-T	1.19	1.18
24	BM25 + Syonyms	QpD-TA	1.12	1.17
25	BM25 + 3 Rewriters	QpD-T	1.22	1.19
26	BM25 + 3 Rewriters	QpD-TA	1.15	1.19
27	BM25 + Twitter IDF	QpD-T	1.21	1.14
28	BM25 + Twitter IDF	QpD-TA	1.06	1.07
29	L2R		1.33	1.36
30	Twitter replies + L2R + Fixed Rule	1 hour	1.45	1.52
31	Twitter replies + L2R + Fixed Rule	1 day	1.53	1.53
32	Twitter replies + L2R + Fixed Rule	1 year	1.52	1.58
33	Twitter replies + L2R + Classifier	1 hour	1.50	1.54
34	Twitter replies + L2R + Classifier	1 day	1.59	1.55
35	Twitter replies + L2R + Classifier	1 year	**1.60**	**1.59**
36	Twitter replies + L2R + Oracle	1 hour	1.54	1.64
37	Twitter replies + L2R + Oracle	1 day	1.61	1.66
38	Twitter replies + L2R + Oracle	1 year	1.61	1.70
39	Oracle		2.13	2.12

pears to be better. However, the significance is observed only on the training set ($p < 0.05$).

None of the query rewriting methods, including their combination, improves the performance significantly, and the same is observed for using the IDF of the Twitter index.

The L2R model is statistically better ($p < 0.05$) than all of the previous configurations, with scores of 1.33 (training) and 1.36 (test).

The hard-coded method for combining Twitter and L2R models (i.e., always preferring Twitter replies when they exist) performs significantly better than all of the previous configurations ($p < 0.05$). Further improvements are observed with the classification-based combination (lines 33 to 35), but with significance seen only on the training set ($p < 0.05$).

To summarize our strongest findings about search strategies for answering aqweets, it appears that the simplest good search approach is applying BM25 scoring on the title field of the crawled Yahoo! Answers. A better learning-to-rank model can be obtained by integrating the output of several retrieval models. Further improvements can be achieved when this L2R model is chosen only when the aqweet receives no replies.

4.4 Answering Live Yahoo! Questions

We study in this section the effect of selecting a particular source of answers, and we examine some approaches for maximizing the answering performance once a source is selected. We do so by describing the architecture of our participating systems in the TREC LiveQA tracks and analyzing their performance.

4.4.1 Answering with Old Yahoo! Answers

In each of the two editions of the TREC LiveQA track we participated with a system that retrieves answers from a crawl of Yahoo! Answers (Section 4.2.1). We index one document per answer. The fields we indexed are the title, the description, their concatenation, and the content of the answer. We tokenize, remove stop words and apply Porter's stemmer [116] with Lucene's English analyzer.

4.4.1.1 A Simple Configuration Selector

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 (Figure 4.8). 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 only in the contents of the subset of best answers. 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 pre-retrieval selection among these alternatives in the first year of the track, we instantiated a small crowdsourcing task on CrowdFlower, in which we showed the annotators questions from a dry run, with up to six answers from the six retrieval configurations (when

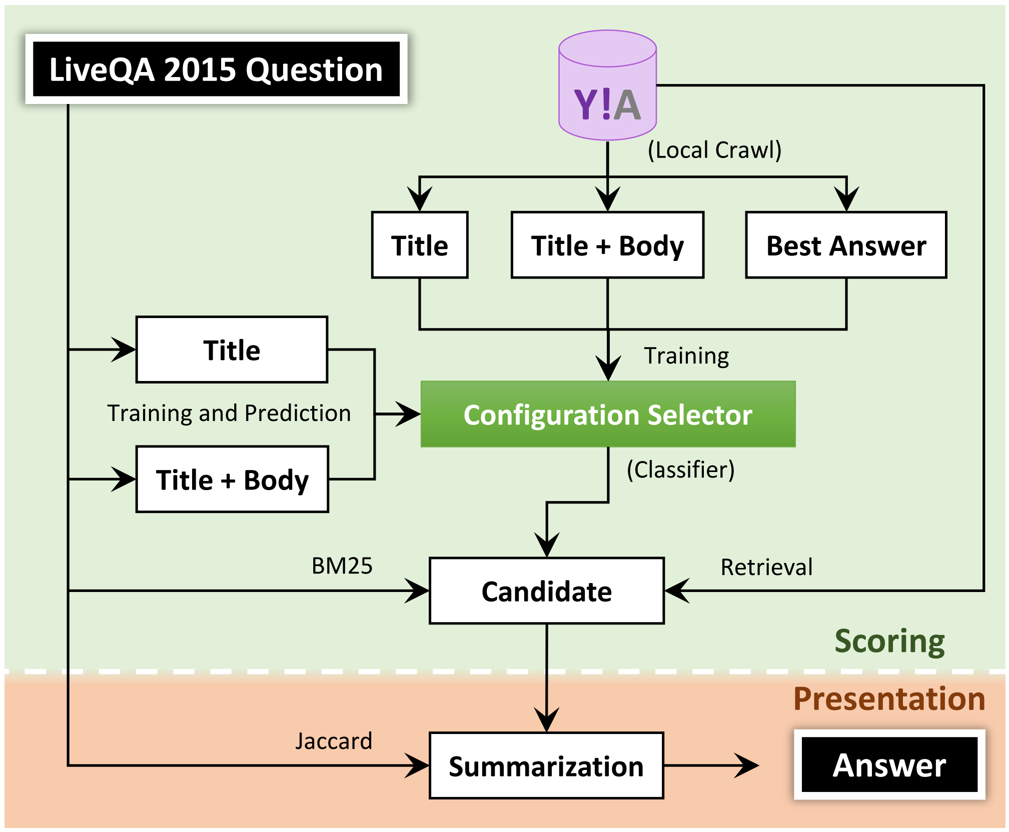


Figure 4.8: Architecture of System CLIP-YA (2015).

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 the annotations of 61 questions assessed by at least three 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[22]

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

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:

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

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.

4.4.1.2 A Cascade of Scoring Functions

An alternative for selecting answers is to defer ranking until after retrieval. Hence, in the second edition of the track, we start by concatenating the title and the description fields of the new question, and we issue the concatenation as a query targeting the fields title, description and answer (i.e., not just best-answer) of the indexed old questions. A list of 100 candidate answers is returned, ranked by Lucene's implementation of BM25.

4.4.1.2.1 Initial Scoring with Old Yahoo! Answers

A retrieval model might be able to find topically relevant answers, but it might fail to identify the good answers among those. Fortunately, we can use our crawl of

Yahoo! Answers to train a classifier to rank topically relevant 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. As illustrated in Figure 4.1, Y!A users can choose up to one best-answer for any given question. They can also vote for different answers by providing thumbs-up and thumbs-down. We define a ranking of the answers for any given question by placing 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 thumbs-up and thumbs-down, 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 (Figure 4.1). When this is the case, we extract the following seven integer features, which might serve as a surrogate for the reputation or the expertise of that user: two values that indicate how active a user is: her level and number of points; the numbers of questions,

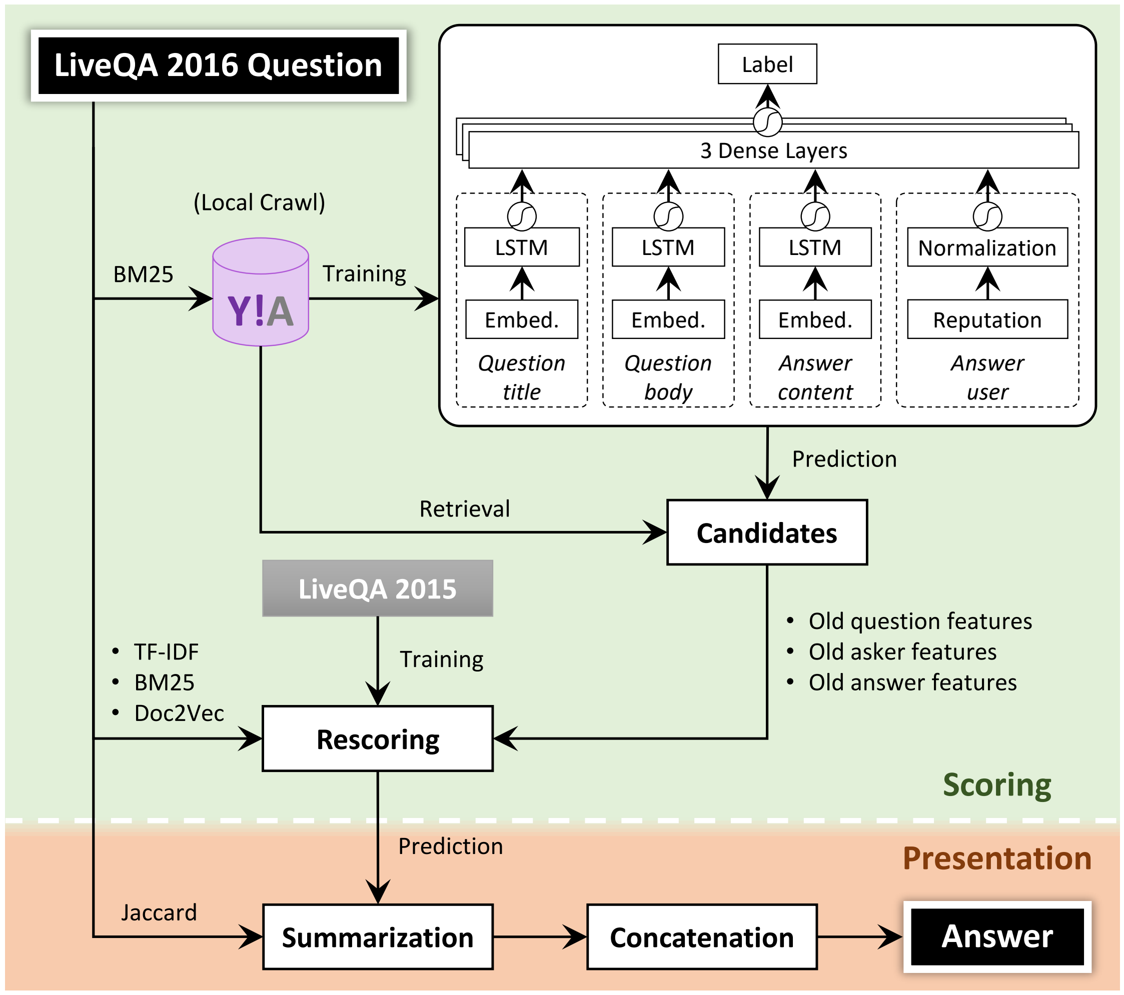


Figure 4.9: Architecture of System CLIP-YA (2016).

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 right corner of Figure 4.9 shows a deep neural network for training on this collection, which we implement in Keras, using Theano as a backend. 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 the best performing system in the first edition of the track [140]). Each of the user features is normalized to a value between 0 and 1, where the scaling 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., the label of the answer). A sigmoid activation is applied between every pair of layers, as well as within the LSTM layers.

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, which we describe next.

4.4.1.2.2 Rescoring with LiveQA 2015 Qrels

The process in Section 4.4.1.2.1 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 the participants of the TREC 2015 LiveQA track returned from Yahoo! Answers, 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 to bring it to the [0-1] range), 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 thumbs-up and thumbs-down, the rating of the answer (a value provided by the asker between 0 and 5, which we divide by 5), and the count of comments that answer received.

- Similarity between the old and new questions: Lucene's implementation of both of TF-IDF and BM25 similarities, and doc2vec cosine similarity (where the document vector is the mean of the vectors of its terms, trained with word2vec [99] on the crawled Y!A corpus) between the title, the body, their concatenation and the answer for the old question from one side, and the title, the body and their concatenation for the new question from the other side (i.e., $3 \times 4 \times 3 = 36$ similarity values).

With the SVM^{rank} software [62], we train a learning-to-rank classifier using the features above, in addition to the score returned by the neural network.

4.4.1.3 Answer Presentation

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. The best performing system from the first edition of the track often combined multiple answers into a single one [140]. This has motivated us, in our participation in TREC LiveQA 2016, 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.

We note, though, that while this approach of "stuffing" answers together might have contributed to improving the performance of our answering system (Section 4.4.3.2), it has also introduced a limitation on the evaluation. In particular, when a good label is assigned to a concatenation of two or more answers, we do not know which answers should get the credit of that label.

4.4.2 Answering from Twitter

Our second source for answering TREC LiveQA questions is Twitter (Figure 4.10). The collection of tweets in which we search is the union of the *SampleStream* (Section 4.2.2.1), *TrackedWords* (Section 4.2.2.2) and *RecentTweets* (Section 4.2.2.4), which we further split into questions, for which we return their replies, and non-questions (Section 4.2.2).

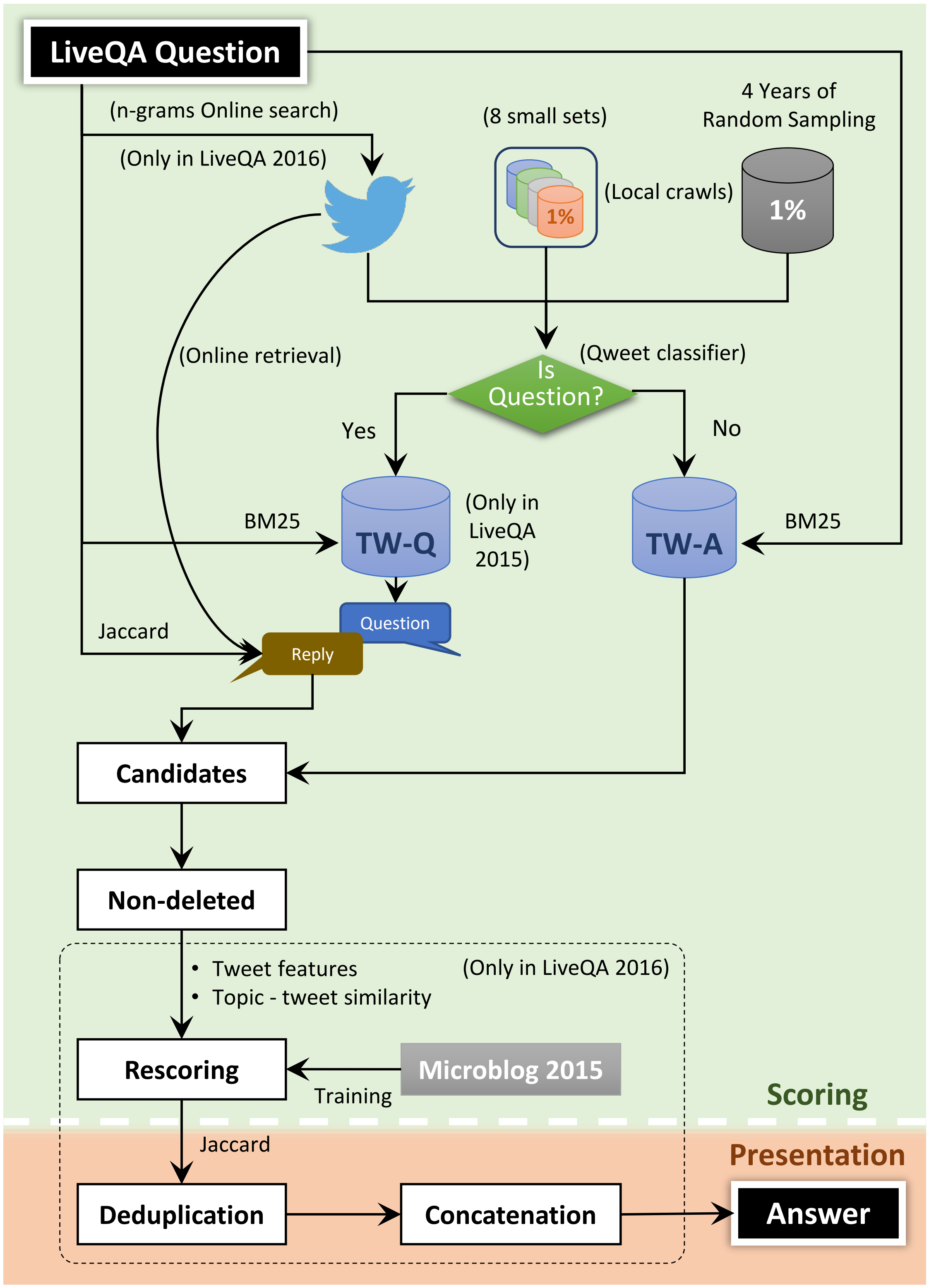


Figure 4.10: Architecture of Systems CLIP-TW.

4.4.2.1 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 their concatenation risks generating a high disk input/output load, and thus exceeding the limit of one minute per question. In our participation in the TREC 2015 LiveQA track, we mitigated this limitation by heuristically selecting 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 our Yahoo! Answers crawl (Section 4.2.1).

2. We issue the preprocessed title as a query to the subcorpus of the local index corresponding to the category of the question, using the 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 the local search engine.

In our participation in the TREC 2016 LiveQA track, we avoided the risk of timing out by placing 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 with no risk.

4.4.2.2 Rescoring with TREC 2015 Microblog Models

In the first edition of the LiveQA track, we relied only on the BM25 scoring function to rank the tweets. In the second year, we enhanced our scoring with a learning-to-rank (L2R) model. With a limited number of good Twitter-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 [84]. The topics of that track contain three fields (e.g., Table 4.1): a short title (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.

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

4.4.2.3 Answer Generation

In our first participation in the LiveQA track, we returned a single tweet as a final answer to the incoming question. Observing the low performance of this approach (Section 4.4.3.2), 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 [6]). 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.

4.4.3 Evaluation

We present some systems that participated in the TREC LiveQA track and compare their performance.

4.4.3.1 Systems

We report and compare the performances of the following systems:

- System **CLIP-YA** (2015) uses a simple classifier to select the fields to use for search in a crawl of Yahoo! Answers. The top-1 result is summarized before being returned (Figure 4.8).

- System **CLIP-YA** (2016) starts by retrieving a list of 100 candidate answers using all the fields of the new question and those of the indexed local crawl of Yahoo! Answers. An initial scoring stage is applied using a deep neural network trained on the Yahoo! Answers crawl. A second rescoring stage is applied using a learning-to-rank classifier trained on the TREC 2015 LiveQA qrels. The rescored answers are summarized and concatenated until the maximum length is reached. This synthetic answer is the final output (Figure 4.9).

- Only 642 out of the 1,015 answers that System **CLIP-YA** (2016) returned were assessed, for reasons of which we are not aware. Thus, we report the scores of a fictive System ***CLIP-YA**** (2016), for which we computed an expected score by multiplying the official score by the ratio of the total number of answers to the number of answers that were annotated: $1.58 = 1,015/642$.

- System **CLIP-TW-A** (2015) searches in the crawled tweets (i.e., both of *SampleStream* and *TrackedWords*) predicted to be non questions. The top scored tweet that has not been deleted is returned (Figure 4.10).

- System **CLIP-TW-Q** (2015) searches in the crawled tweets (i.e., both of *SampleStream* and *TrackedWords*) predicted to be questions. Among the replies to the top 20 scored tweets that have not been deleted, we return the one with the highest Jaccard similarity (Figure 4.10).

- System **CLIP-TW-A** (2016) searches in the crawled tweets,[23] as well as in the recent tweets returned by Twitter's Search API (i.e., *RecentTweets*), with

[23]We had a particular focus on the *Travel* category, hence we tracked the words of its 27 subcategories as well.

a restriction to tweets predicted to be non questions. The top scored tweets (for which the exact number depends on the number of tweets returned by the Search API) are rescored with a learning-to-rank model trained on the TREC 2015 Microblog qrels. The resulting ranked list is deduplicated with the single-link clustering algorithm using Jaccard as a similarity measure, and concatenated up to the allowed limit of 1,000 characters. Unfortunately, none of the answers returned by this system was assessed. While we do not know the exact reason, we have found that Java's default XML parser has a known bug prohibiting it from loading XML files with emojis.[24] 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. Thus, we report the scores of a fictive System ***CLIP-TW-A**** (2016), for which we computed an expected score using the method described in Chapter 5.

- **CMU-OAQA** is a system from the Carnegie Mellon University team [140, 141]. It uses a deep neural network and synthesizes one answer or more into a single one. This system has the highest official score over all automatic runs in both editions of the LiveQA track.

- **Emory-Crowd** is a hybrid system from the Emory University team [123], where the crowd workers participated in the selection of returned answers. This system has the highest score among the teams that participated in the second edition of the LiveQA track.

[24] http://stackoverflow.com/questions/31867818

Table 4.4: Effectiveness of participating systems in the LiveQA task. In 2016 only, the average of all runs includes manual runs.

Year	System	Answers	Score
2016	Human-QUAL	778	1.561
2016	Human-SPEED	849	1.440
2016	*CLIP-YA**	1,015	1.344
2016	Emory-Crowd	976	1.260
2016	CMU-OAQA	954	1.155
2016	CLIP-YA	642	0.850
2016	*CLIP-TW-A**	1,015	0.846
2016	Mean(all runs)	774	0.643
2015	CMU-OAQA	1,064	1.081
2015	CLIP-YA	1,079	0.615
2015	CLIP-TW-A	805	0.144
2015	CLIP-TW-Q	1,066	0.081
2015	Mean(all runs)	1,007	0.465

- **Human-SPEED** and **Human-QUAL** are two fictive systems that the organizers of the second edition of the LiveQA track created by crawling the answers posted on the question Web page one week after the evaluation took place. The former uses the first answer submitted to each question, while the second uses the best answer selected by the asker, if one exists, or is selected by "Yahoo's quality scoring algorithm" [2].

- We also report the average scores of all participating runs (including the human and hybrid systems, when applicable).

4.4.3.2 Overall Effectiveness

In Table 4.4, we compare the effectiveness of our systems with the best performing systems in the two editions of the LiveQA track. Looking at Twitter based systems, both performed very badly in 2015 (with scores below average). But between the two, **CLIP-TW-A** is better. This encouraged us to drop **CLIP-TW-Q** in 2016, focusing on the improvement of **CLIP-TW-A**. Using our score estimator (see Chapter 5), it appears that our modifications to this system were fruitful, increasing its score from 0.144 (below average) to 0.846 (above average).[25] Our use of a SSD also helped us to avoid timing out.

CLIP-YA is our best system in both years. The two scoring stages, combined with the answer synthesis step contributed to doubling its score from 0.615 in 2015 to an estimated score of 1.344 in 2016. Interestingly, this system, in its second year, performed better than the hybrid system, and not very far away from the human runs. The comparison against the latter systems is, of course, not fair, given that the number of answers provided by the human assessors is less than those returned by ***CLIP-YA****. However, those unanswered questions suggest that combining our automatic system with human answers might lead to a higher user satisfaction. One way for combining them can be by training a classifier in a manner similar to what we have shown in Section 4.3.7 for answering Twitter questions.

In the remainder of this section, we look at the results from different perspectives, with a focus on **CLIP-YA** and **CLIP-TW-A** (for both years).

4.4.3.3 Scores per Category

The questions of the TREC LiveQA track are not equally distributed across their categories. This distribution has an indirect impact on the aggregate perfor-

[25]We assume that the topics maintain their difficulty between the two years and, consequently, that a comparison across the two years is meaningful.

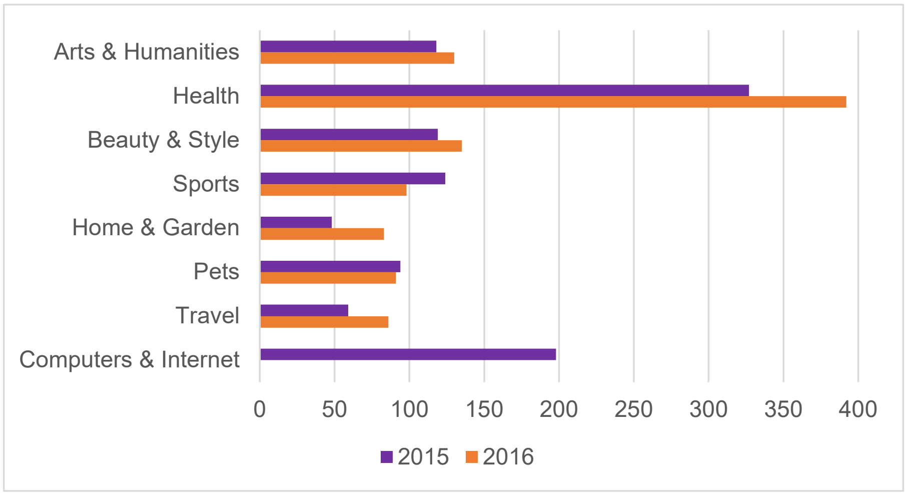


Figure 4.11: Distribution of questions across the TREC LiveQA track categories.

mance of the systems (e.g., it is perhaps more important to improve the accuracy on popular categories). As Figure 4.11 indicates, there is a clear dominance of the *Health* category, which accounts for about one third of the questions, followed by *Computers & Internet*, corresponding to about one fifth of the questions in the first edition of the track. Next to that, we have *Arts & Humanities*, *Beauty & Style*, and *Sports* that, each, matches about one tenth of the questions. Each of the remaining three categories correspond to between 5% and 9% of all questions.

The performance of our systems also vary per category, with a general tendency towards demonstrating good results on popular categories. In Table 4.5, we show the improvement of the scores across two years for each category and system. We also indicate the size of the documents in our crawl used for retrieval in the second year (i.e., number of question/answer pairs for **CLIP-YA**, and tweets count for **CLIP-TW-A**). This table confirms the superiority of **CLIP-YA** over **CLIP-TW-A**, but we note that their performance is comparable on the *Travel* category, which appears to be the most difficult topic for the former system in both years, and for

Table 4.5: Average scores of our LiveQA systems per category, with the relative improvement between two years, and the size of the indexed corpus.

Category	#Leaves	CLIP-YA[*]				CLIP-TW-A[*]			
		2015	2016	Imprv.	Size	2015	2016	Imprv.	Size
Arts & Humanit.	14	0.64	1.37	+114%	17M	0.16	0.70	+337%	45M
Health	21	0.77	1.50	+114%	41M	0.20	0.85	+305%	43M
Beauty & Style	6	0.50	1.60	+220%	33M	0.21	0.87	+314%	42M
Sports	56	0.51	0.98	+92%	31M	0.23	0.80	+248%	43M
Home & Garden	6	0.60	0.96	+60%	5M	0.09	0.81	+800%	69M
Pets	8	0.65	1.55	+138%	22M	0.17	0.95	+459%	36M
Travel	368	0.29	0.63	+117%	9M	0.25	0.65	+160%	748M
Computers & Int.	20	0.60	-	-	-	-[26]	-	-	-

the latter system in the second year. Interestingly, that category was the easiest one for **CLIP-TW-A** in the first edition of the track (which has encouraged us to gather more tweets for it in the second year).

We observe a moderate correlation (Pearson's $\rho = 0.63$) between the performance of ***CLIP-YA**** in 2016, and the size of its retrieval corpora. This has two potential explanations. With a large crawl, (1) we are perhaps more likely to cover new questions, and (2) we have more training data for our initial scoring stage (Section 4.4.1.2.1).

There is an additional factor that impacts the differences in the performance of our ***CLIP-YA**** system across the seven categories. We looked at the number of leaf categories in the hierarchies of those parent categories, and found a large variance between them. For instance, the most difficult category (i.e., *Travel*), has a large number of leaves (i.e., 368). An intuitive potential explanation is that almost identical questions about two different places (i.e., from two different sub-

[26]Due to a corruption in our index of the *Computers & Internet category*, **CLIP-TW-A** missed answering all questions in that category.

Table 4.6: Number of questions answered by each of our TREC-2015 LiveQA systems 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
- classifier decision or risk timeout	0.68	681	0.08	26	0.19	98

categories), might require very different answers. Over the seven categories, we found a moderate negative correlation ($\rho = -0.75$) between our performance on one category, and the number of its leaf nodes. Finally, we combined the size of the crawled category, and the number of leaves in the function below, and found it has a high statistically significant positive correlation with the performance of ***CLIP-YA**** ($\rho = 0.87$, $p < 0.05$):

$$f(category) = \frac{log\,(size(category))}{leaves(category)}.$$

4.4.3.4 Using the Body of the Question

Our 2015 systems have different strategies to decide whether to use the terms that appear in the body of the question for retrieving answers (Section 4.4.1.1). 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 4.6). 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 4.7: Number of questions answered by CLIP-YA (2015) depending on the retrieval field, with the corresponding score.

	CLIP-YA (2015)	
	Score	Count
Question title	0.43	109
Question title and body	0.54	186
Answer	0.67	784

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 more than half of the allowed response period. These two systems are configured not to 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.

4.4.3.5 Retrieval Field for Old Yahoo! Answers

The classifier used by System **CLIP-YA** (2015) chooses between three configurations for the fields to be searched in the old answers. As shown in Table 4.7, 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 (0.67) higher than those for which the classifier chose to search in the content of the old questions. Among the latter cases, the average score when the body of the old question was included in the search is higher than the average score when it was not included (0.54 vs 0.43, respectively). Overall, it appears that the more content we search in, the better the result we can expect. Consequently, we decided, in the second edition of the LiveQA track, to search in the combination of the title, body, and answer of old answers, for each incoming question.

4.4.3.6 Best CLIP-YA Configuration

Combining these insights, we might speculate that the best configuration of our **CLIP-YA** (2015) system would be one that uses the title and the body of the incoming question as a query (Section 4.4.3.4), and the index of old answers for retrieval (Section 4.4.3.5). As it happens, only three questions were answered 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, which justifies our inclusion of all text fields of the incoming questions and the crawled question/answer pairs in the second year.

4.4.3.7 Effect of Twitter Retrieval Corpus

Systems **CLIP-TW** (2015) retrieve answers from Twitter using 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 4.8 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. We, therefore, collected more tweets with

Table 4.8: Number of questions answered by the CLIP-TW (2015) systems 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

this method, on which ***CLIP-TW-A**** (2016) was evaluated.

4.4.4 On Combining Answers from Twitter and Yahoo! Answers

Our TREC LiveQA systems were running in parallel independently from each other. Whereas, on average, **CLIP-YA** is better than each of the **CLIP-TW-A** and **CLIP-TW-Q** systems, it might be the case that each of them excels at different questions. To assess the extent to which that might be the case, we examine the effectiveness we can obtain if an oracle were to tell us us which system to interrogate for which question. We consider only the systems **CLIP-YA** and **CLIP-TW-A** because they had the highest scores among our three systems in the first edition of the track and because these are the only systems we used in our participation in the second year.

To focus our analysis, we limit ourselves to the 805 questions that were answered by both systems in 2015, and to the 498 questions for which (1) we obtained official annotations in 2016 (only for system **CLIP-YA**), and (2) at least one answer from some participating system (including human systems) was assessed as excellent.[27] We group the questions by Yahoo! Answers pre-defined question category

[27]Our relevance estimator that we apply to estimate the effectiveness of **CLIP-TW-A** requires

Table 4.9: Expected scores from an oracular selection between the answers of **CLIP-YA** and **CLIP-TW-A**.

Category	Year	#Ques.	CLIP-YA	CLIP-TW-A	Oracle	Improv.
Arts & Humanities	2015	111	0.66	0.16	0.75	13.70%
	2016	61	1.66	0.72	1.72	3.68%
Health	2015	293	0.81	0.20	0.86	6.78%
	2016	220	1.65	0.88	1.76	6.86%
Beauty & Style	2015	107	0.50	0.21	0.60	18.52%
	2016	47	1.81	0.87	1.90	4.85%
Sports	2015	115	0.52	0.23	0.64	23.33%
	2016	32	1.16	0.82	1.51	30.61%
Home & Garden	2015	44	0.64	0.09	0.66	3.57%
	2016	33	1.30	0.81	1.47	13.09%
Pets	2015	82	0.72	0.17	0.79	10.17%
	2016	65	1.80	0.99	1.98	9.85%
Travel	2015	53	0.28	0.25	0.47	66.67%
	2016	40	0.83	0.65	1.07	29.34%
All	2015	805	0.65	0.20	0.74	12.76%
	2016	498	1.56	0.85	1.71	9.10%

and compute (or estimate) the average score of both systems per category. Finally, we calculate the score of the oracle by selecting the maximum score between the two systems for each question, and then computing the average over each question category.

Table 4.9 shows the results, including the relative potential improvement compared to using the average best system only (which is not the best system for each individual question). We can get about 10% improvement over all of the questions. But the relative improvements differ per category. In particular, those potential improvements are higher for categories in which **CLIP-YA** scores poorly. For each edition of the track, we computed the Pearson correlation between the average score

that at least one labeled answer has a perfect score (Chapter 5).

of **CLIP-YA** in each category, and the potential relative improvement. We found a strong negative statistically significant correlation in each year ($\rho = -0.89$ and $p < 0.01$ for both of them). This sets a clear direction for future work on combining answers from Twitter and Yahoo! Answers on categories for which **CLIP-YA** has the least performance, which are *Travel* and *Sports*. We note that this agrees with our earlier observation about the diversity of the questions, in which those same categories had the highest number of leaf categories (Section 4.4.3.3).

4.5 Deciding to Answer

The TREC RTS track (Section 4.1.2) allows us to study a variant of the problem of deciding whether to return an answer to a question. In this section, we describe a system that makes a real-time decision for returning a novel interesting tweet to a user, and analyze its performance.

4.5.1 Components

Our main components for this task are a relevance model, a learning-to-rank stage, and a novelty detector (Figure 4.12).

4.5.1.1 Relevance Models

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 (e.g., Figure 4.1). We stem the topic fields

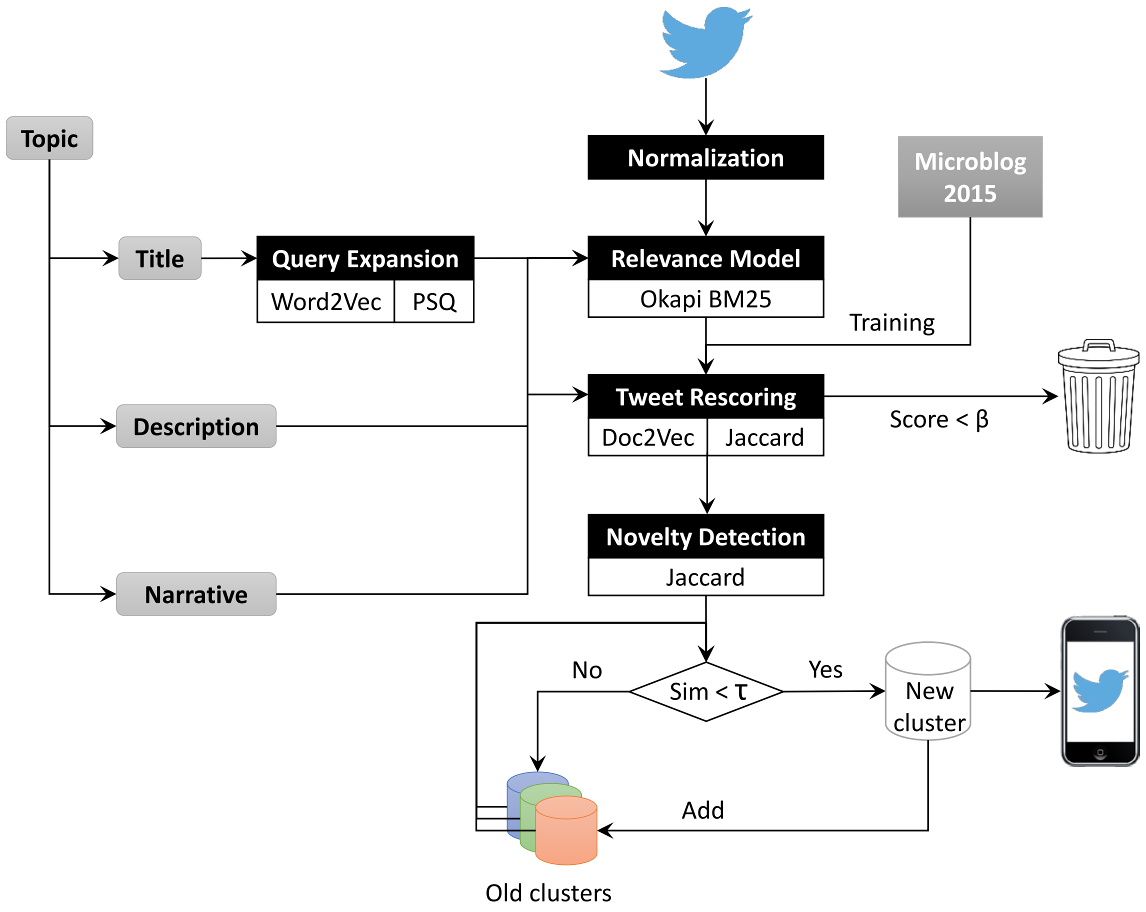


Figure 4.12: Architecture of our TREC 2016 RTS Systems.

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 by removing emoticons, user mentions, URLs, RT indicators, and punctuation, before stemming.

Our relevance models are based on Okapi BM25 term weights and title expansion using word embeddings and probabilistic structured queries [34]. We use the tweets of the *SampleStream* corpus (Section 4.2.2.1) to train a word2vec model [99] 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 cosine similarity 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:

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

We obtain the scores for the description and narrative fields by applying the BM25 model without expansion. That is, for a stemmed term t of either field:

$$Score(d, Q_{Field}) = \sum_{t \in Field} BM25(TF(t, d), DF(t)).$$

We tune the parameters using a grid search on the TREC 2015 Microblog track topics. We set k1 = 0.09, b = 0.5 and average document length = 21 for BM25, and $J = 5$.

4.5.1.2 Tweet Rescoring

To refine the scores of the relevant tweets, we use the SVM^{rank} package [62] to train a learning-to-rank model based on the TREC 2015 Microblog track topics, using the relevance scores of the previous stage (Section 4.5.1.1), in addition to the following features:

- **Sender popularity 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 Jaccard similarities (proposed by Magdy et al. [93], in which the denominator of Jaccard index is either the minimum or the maximum size of the compared bag-of-word pairs, instead of the size of their union) between the tweet and each of the title and description fields of the topic, and the cosine similarity between the doc2vec vector (i.e., the mean of the word2vec vectors) of the stemmed terms of the tweet and the doc2vec vectors of each of the description and narrative fields of the topic.

Tweets that have a (re-scored) score less than a threshold β (set manually based on our participation in the TREC 2015 Microblog track [6]) are eliminated at the end of this stage. The remaining tweets go to the novelty detection phase.

4.5.1.3 Novelty Detection

According to the TREC RTS 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 that has not been discarded in the rescoring stage, we assign the tweet to the cluster containing the most similar tweet, if the similarity exceeds certain manually selected threshold τ (described below). 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. The tweet is pushed to the user as soon as a new cluster is created, and 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).

4.5.2 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.[28] If no tweet is

[28]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]`.

returned, we consider the union of tweets returned from issuing subqueries in which one term of the title was removed. We score all of the returned tweets as explained in Section 4.5.1.2. 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.

4.5.3 Evaluation Metrics

Two distinct evaluation setups were introduced for the push scenario task of the TREC 2016 Real-Time Summarization track. In the first setup, the tweets returned by participating systems are communicated to a "broker," which then sends them to real users (paid students) as push notifications on their smartphones. Those users might decide to ignore the tweet, or judge it as relevant, redundant (i.e., the relevant content has been communicated earlier in another tweet), or not relevant. Two precision measures were reported based on this framework using the judged tweets. They differ on whether redundant tweets are considered to be relevant. We adopt the precision measure that penalizes redundant tweets (**P(strict)**), and we note that the ranking of our systems versus the other systems does not change if we alternate the precision measure.

The second evaluation setup is based on standard batch evaluation by TREC assessors. For each interest profile, a pool of tweets was first constructed from the submitted runs (including the "email digest" scenario—not of our interest in this dissertation). Next, those tweets were judged as not relevant (gain=0), relevant (gain=0.5) or highly relevant (gain=1). The relevant and highly relevant tweets were then grouped into semantic clusters to indicate redundancy. If a system retrieves more than one relevant tweet from the same cluster, then only one of them is considered to be relevant. Among the seven measures that were reported for the tweets returned by a system for an interest profile on a particular day, we focus on one variant of the *Expected Gain (EG)*, which is generally defined as the sum of the gain divided by the number of returned tweets. The variant of expected gain on which we focus, **EG-1**, gives full credit on a silent day for systems that returned nothing, and a score of 0 if they returned anything. We chose EG-1 because it was used for the official rank of systems, and because it takes in consideration the decisions made by participating systems during silent days.

4.5.4 Results and Analysis

We participated with three systems in this task. **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 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. This arbitrary threshold of 0 was chosen to be sufficiently lower than all of the per-topic minimum thresholds (for which the values range between 4.38 and 10.43).

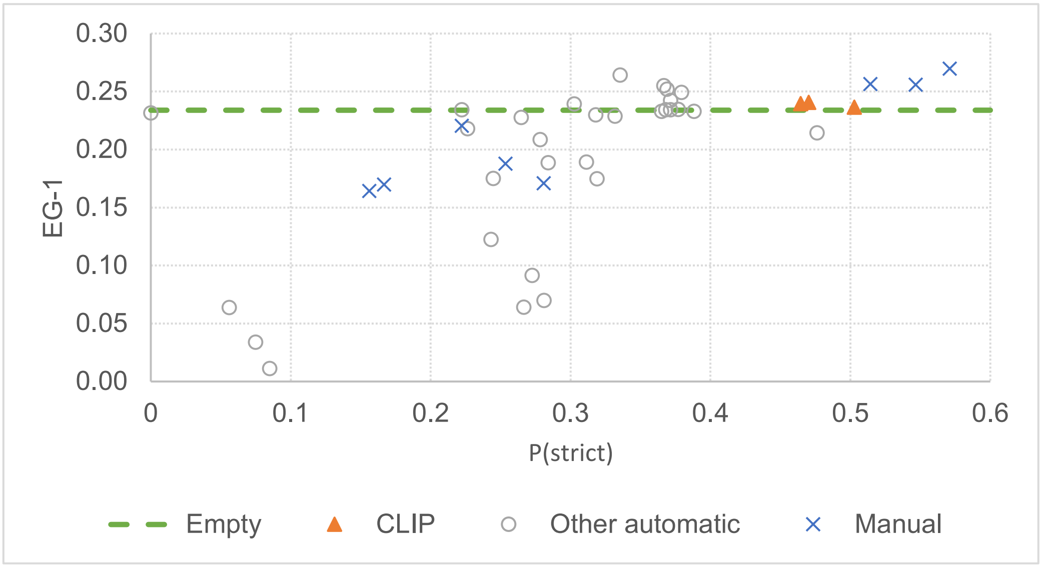


Figure 4.13: Comparison of RTS official scores between all participating systems.

We compare our systems against four other systems. **PolyU-run3** is the best system based on each of the measures P(strict) and EG-1. The Hong Kong Polytechnic University team, who developed this system, made a manual intervention to select query terms [133]. **QUBaseline** is the best automatic system based on the EG-1 measure, and was developed at Qatar University [131]. **UmdHcilBaseline** is the only automatic system that had a P(strict) value higher than some of our systems. It was developed by the HCIL team at the University of Maryland independently from ours [85]. **Empty-run** is a fictive system that returns nothing.

Figure 4.13 compares the performance of our systems against all other systems on both measures. We note, first, that the empty run is a strong baseline for EG-1. In fact, only 13 (including all ours) out of 34 automatic runs managed to get a higher score. This phenomenon is due, to some extent, to an arbitrary choice made for the definition of this measure. EG-1 penalizes equally a silent system on a verbose day (i.e., false negatives), and a verbose system on a silent day (false positives). In a real application, a user might be more tolerant of one error versus another. For instance, she might not be bothered much by a single interruption on a silent day, as long as relevant novel tweets are fully covered on other days. Another critique of this measure is that it gives the same penalty, on a silent day, for a system that returned only 1 tweet and another one that returned 10 tweets. We expect a user to be more displeased with the second system. Hence, the penalty in a better measure should reflect the degree of verbosity of systems on silent days.

We observe four main clusters of systems that outperformed the empty run on EG-1. Three manual runs (all from the same team) are located at the top right corner of the scatter plot. One of them (**PolyU-run3**) scores the best on both measures (see Table 4.10). Our systems, by contrast, barely exceed the performance of the empty run on EG-1. However, our strict precision is better than most of the other automatic runs by a large margin.[29] The only automatic system that has a strict precision higher than two of our systems (**UmdHcilBaseline**) has an EG-1 below that of the empty run. The best automatic run on EG-1 (**QUBaseline**) has a relatively low strict precision (0.3), and appears to be isolated as a third cluster. The fourth group of runs is clustered just above the empty run's threshold with a strict precision less than 0.4.

The detailed differences between our own systems are indicated in Table 4.10. System **CLIP-A-0.7-MAX** is the only automatic system that has a strict precision above 0.5. This is perhaps due to its high clustering and relevance thresholds,

[29] We do not have access to the complete individual runs of other systems to run statistical significance tests.

Table 4.10: Effectiveness of participating systems in Scenario A of the RTS track.

	Mobile assessors					NIST
System	relev.	redund.	not rel.	unjudged	P(strict)	EG-1
CLIP-A-0.7-MAX	91	1	89	507	0.5028	0.2366
CLIP-A-0.5-MEAN	158	7	171	911	0.4702	0.2407
CLIP-A-0.5-0	170	7	189	1,071	0.4645	0.2397
PolyU-run3 (manual)	193	4	141	1243	0.5710	0.2698
QUBaseline	56	3	108	477	0.3353	0.2643
UmdHcilBaseline	20	0	22	176	0.4762	0.2145
Empty-run	0	0	0	0	-	0.2339

compared to **CLIP-A-0.5-MEAN** and **CLIP-A-0.5-0**. For instance, the 6.93% relative improvement in P(strict) between **CLIP-A-0.5-MEAN** and **CLIP-A-0.7-MAX** came at a small relative loss of 1.70% in EG-1.

Although there is no official recall measure reported, perhaps because the number of tweets assessed for each system is different, we observe that the number of relevant tweets we returned is substantially larger than that of the two automatic systems **QUBaseline** and **UmdHcilBaseline**. Thus, our high precision was obtained with relatively high recall as well (the empty run might be considered to have a perfect precision of 1.0 but its recall would be zero). It appears that both our scoring of the tweets, and the cutoff point were relatively good for this task.

4.6 Chapter Summary

We studied in this chapter several aspects for providing answers within and cross platforms. For questions asked on Twitter, we found that user satisfaction can be improved substantially if we return a thread from Yahoo! Answers that has a similar question in the title. Further improvements are obtained with a learning-to-rank model, especially if the retrieved threads are combined with the replies that the original question receives on Twitter. For questions asked on Yahoo! Answers, we found that a two stages scoring approach, based on answers retrieved from a large crawl of old Yahoo! Answers, advances the state of the art on answering live complex questions, as organized in the TREC LiveQA track. Seeking answers in another platform, we found that Twitter can be particularly useful for categories of questions that are difficult to address using old Yahoo! Answers. Finally, we examined the problem of deciding whether to return tweets to some interest profile. We introduced a system that makes such a decision based on statistics from old tweets. This system had the highest precision in the TREC RTS track.

Developing better answering systems requires the ability to assess their effectiveness. The setup of the TREC LiveQA track evaluation makes the reusability of the released test collection non trivial. Next chapter discusses this challenge and proposes some adequate solutions.

Chapter 5: Evaluating Future Answers[1]

Modern information retrieval test collections have proven to be remarkably useful as a basis for rapidly and affordably comparing alternative retrieval algorithms. In recent years, this capability has also been leveraged to learn parameter settings that optimize specific evaluation measures, an approach now commonly called learning to rank [87]. In large-scale systems, these "offline" evaluations typically serve as initial triage before subjecting well tuned systems to further testing (e.g., A-B testing or interleaving) with live users. Creating test collections that can be used in this way can, however, be an expensive undertaking. The fundamental challenge is that to evaluate a system we must know the relevance of the items (i.e., documents or answers) that it finds, but to know which items good systems will find we must already have good systems. The usual approach is therefore iterative. For example, first-year test collections for a new task that are created by TREC typically only include relevance judgments for items found by what will (in future years) be thought of as baseline systems. Second-year test collections are typically built using relevance judgments for samples drawn from the results of better systems, and thus are more useful as a basis for tuning relatively good systems. It is

[1]Some parts of this chapter were taken from a paper under review by Bagdouri and Oard [12].

typically in the third year of a task that systems that are well tuned to a specific evaluation measure can be created.

A good deal of effort has gone into shortening this cycle by developing techniques that can support the requisite system comparisons using less well sampled relevance judgments, an approach we might think of in a shorthand way as seeking to achieve "year 3 results in year 2." Three broad types of approaches have been tried. The first approach was what we might today call "diversity sampling." The goal of this type of diversity sampling is to increase the chance that the judgments performed when the test collection was created will have been performed on some items that ultimately will be retrieved by future systems. The initial approach to diversity sampling was "pooling" – the aggregation of highly ranked items from a diverse range of systems into a pool that would then be exhaustively judged [137]. It was quickly realized that fully automatic systems were often not sufficiently diverse for this purpose, so results from human-in-the-loop systems were also often included in the judgment pools [137]. Early evaluation measures such as Mean Average Precision (MAP) [18] were computed using an implicit or explicit assumption that unjudged items were not relevant. Typically, however, very good future systems will find relevant items that were not found by any early system, and thus were not judged. This led to the development of measures such as bpref [19] and xinfAP [150] that could be used to more reliably compare systems despite a relative paucity of judgments for items retrieved by those systems. Of course, this approach is fundamentally limited since it requires that judgments be available for at least some of the items that future systems will retrieve.

Initial retrieval tasks focused on retrieval of documents. However, retrieving parts of documents (e.g., passages or facts) magnifies the combinatorial complexity of the sampling process. Moreover, the introduction of tasks such as the TREC-2015 LiveQA track [1], in which the items to be judged might be automatically constructed from parts of multiple documents, and in which future answers might be constructed from documents that did not exist (or were not known) at the time the initial judgments were performed, moves us beyond any hope of being able to rely on pre-constructed judgments alone. To address such cases, a third approach based on automatically estimating what judgment a human assessor would have made if presented with a newly retrieved item was developed [24].

One challenge with evaluation approaches based on estimated relevance judgments is that the systems that we are seeking to evaluate are themselves algorithms for estimating relevance judgments. This leads to a potential circularity in which we are in essence using one retrieval system to evaluate another. This only makes sense when the estimation approach that we are using as a basis for evaluation has more information available about the likely relevance of an item than do the systems being evaluated. Approaches to evaluation that are based on estimated relevance are thus naturally, and necessarily, based in the first instance on diversity sampling. Evaluation measures that are designed to accommodate unjudged items in effect (although not always explicitly) implement a restricted form of relevance estimation, so a well designed approach to relevance estimation may not benefit further from the use of such measures.

We introduce two new relevance estimation techniques based on word embeddings. They both achieve excellent reusability results for the TREC 2015 LiveQA track, and the best of them also demonstrates a good reusability with the TREC-8 Ad Hoc document retrieval task [138]. Our third, and best, technique also leverages an approach to diversity sampling, improving our relevance estimates. We compare our results with three increasingly complex baselines.

The dominant paradigm for evaluating reusability has been to ablate a single system and then to estimate what retrieval effectiveness score (e.g., NDCG or MAP) that ablated system would have achieved given some way of combining one or more of diversity sampling, relevance estimation, and a sparsity-tolerant evaluation measure. We therefore report results in that way for comparability with earlier work. We also report results using an improvement to that approach in which we learn a regression function to map estimated scores to the same score space as the unablated systems. We characterize the resulting stability or instability of system rankings using a new correlation coefficient that was developed specifically to code the ranking of best systems and the gap between their scores [47].

One limitation of the single-system-ablation approach is that it is necessarily focused on mean values across many queries, but it is insensitive to changes in the estimated variance. We therefore also adopt an approach similar to that proposed by Moffat et al. [102] and Jones et al. [64] in which we characterize the effect of system ablation on the results of a statistical significance test (in our case, the *t*-test). Because these tests are conducted on paired system samples, we use paired ablation in that case. All three approaches to evaluating reusability yield results that are broadly consistent, and they specifically agree on the preference order among the three methods for enhancing reusability using word embeddings (in one case together with diversity sampling) that we introduce in this chapter.

The remainder of this chapter is organized as follows. Section 5.1 introduces three baselines and three novel relevance estimation methods we use to enhance reusability. That's followed by Section 5.2 in which we introduce our two novel approaches to assessing reusability. Section 5.3 then evaluates the six methods using both the standard single-ablation approach and our two novel alternatives based on the TREC-2015 LiveQA track runs. Section 5.4 briefly explores the generalizability of our findings to the TREC-8 Ad Hoc runs. We summarize our observations in Section 5.5.

5.1 Relevance Estimation Methods

Our goal is to predict the performance of a system that did not participate in the TREC LiveQA track without requiring new annotations. Our intuition is that by looking at the content of unlabeled "system candidate" items, and comparing them to the content of labeled "reference" items, we can approximate the relevance judgments that would have been assigned by TREC assessors to items returned by future (i.e., non-participating) systems.[2] In this section, we introduce three baselines and three novel methods for performing this approximation. All of these methods

[2]In LiveQA the items are answers; in the TREC-8 Ad Hoc task the items are documents; for generality we refer to items when discussing methods that could be used at either scope. Similarly, we use "topics" as an inclusive term when we don't wish to distinguish between questions in the LiveQA setting and queries in the TREC Ad Hoc setting.

operate on stemmed unigrams after removing stop words using the default English stemmer of Lucene 6.4.2. They take as input an unlabeled item d and a set of reference (i.e., labeled) items R. They produce an unbounded estimated relevance score $EsRel(d, R) \in]-\infty, +\infty[$. When we wish to produce an estimated relevance score that is comparable to the official ones (i.e., between 0 and 3 for TREC LiveQA; and between 0 and 1 for TREC-8 Ad Hoc), we apply the relevance estimator to each reference item r, producing a set of scores $EsRel(r, R)$. We perform a linear regression that maps these scores to the official ones. We project the original score $EsRel(d, R)$ with this linear fit. We then bound the values outside the allowed interval to one of the edges of this interval (e.g., for TREC LiveQA, we clip the scores to the $[0, 3]$ interval). We denote by this final projected estimated relevance $\overline{EsRel(d, R)}$.

5.1.1 Item Length

A simple baseline is the item length, which we express as the count of the terms in the unlabeled item. The reference items are ignored in the calculation of the unbounded estimated relevance score. But they are used for computing the projected one.

$$EsRel_{TC}(d, R) = \sum_{t \in d} count(t, d)$$

The utility of this feature arises because neither of the evaluation measures for either of the two test collections that we use in this chapter reward greater brevity. We therefore include item length as a simple relevance estimator.

5.1.2 Clipped Term Counts

An improvement over item length is clipped term counts, which was introduced by Papineni et al. as a first step in a series of computations to calculate the BLEU score for machine translation evaluation [111]. BLEU is based on the overlap between the output of a system and a set of reference translations, and clipped term counts are a simple way of preventing any one term from dominating the result. In this way it fills the same role as sublinear transformations such as the logarithm of the term frequency in a vector space term weight function, but with the computational convenience of integer counts. We implement this notion in the following way. We consider a reference item to be any item that has the highest possible relevance score (for TREC-8 Ad Hoc, this is 1; for TREC LiveQA this is 3). Let t be a term in an unlabeled item d, and r be one member of the set of reference items R. We start with the maximum count of t in any reference item. We then clip this value to be no larger than the count of t in the unlabeled item. The unbounded estimated relevance score for the unlabeled item d is then computed as the sum of the clipped counts:

$$EsRel_{CTC}(d, R) = \sum_{t \in d} \min\left(count(t, d), \max_{r \in R}(count(t, r)) \right)$$

We note that this idea of clipped term counts is the only part of the BLEU computation that we use.

5.1.3 Core Vocabulary

Clipped term counts leverage the good reference items, but they ignore the evidence that is available from reference items that are known to be bad. So for our third baseline approach, we would like to reduce the impact of terms that appear in both good and bad items, and to give more weight to the terms that appear much more often in good reference items than bad ones.

We propose to identify terms that are likely to be correlated with only good items in the following way. For each topic (i.e., query or question), we create two bags of terms. The first, denoted as the positive bag P, is a concatenation of all good items (i.e., those labeled 1 in TREC-8 or 3 in LiveQA). The second is a concatenation of all bad items and is denoted as the negative bag N. For every term that appears in the positive bag, we compute its probability as the ratio of its occurrence to the number of terms in the positive bag. We subtract from this value the probability of the same term in the negative bag. The result is a value that indicates the utility of this term. We restrict ourselves to the terms that have a utility above the average utility of all terms in the positive bag, defining a core vocabulary for that particular topic. Finally, we estimate the relevance of the unlabeled item as the sum of the count of its terms that belong to the core vocabulary, divided by the natural logarithm of one plus the term count of the unlabeled item.

$$EsRel_{CV}(d,R) = \frac{\sum\limits_{t \in d \cap v} count(t,d)}{log\left(1 + \sum\limits_{t \in d} count(t,d)\right)}$$

5.1.4 Vocabulary Expansion

Both our clipped term counts and our core vocabulary baseline relevance estimators rely on an exact match between the terms in the unlabeled and reference items. It might be useful to take into consideration terms that share some similar meanings, even if they are not identical at the surface level. We propose to expand the core vocabulary by expanding the positive and negative bags of terms for the reference items. Assume we have access to a matrix that gives us the similarity between any pair of words. For both the positive or the negative bags, we expand each term with the most similar terms, using the value of this similarity (assumed to be bounded between 0 and 1) as a substitute for the occurrence of the similar term. In all of our experiments in this chapter, we used the word embeddings released by the GloVe project [114]. We downloaded a model pre-trained on 840B words from CommonCrawl,[3] and stemmed the words using Lucene. When there was a collision of terms after stemming, we maintained the vector of the first unstemmed word that appears in the embeddings file. We used cosine between the vector representations of two terms to estimate their similarity, restricting ourselves to the top 10 positive values.[4] Once we have the positive and negative bags expanded, we proceed in the same manner as we do for the core vocabulary method above (Section 5.1.3).

[3] http://commoncrawl.org

[4] Unlike the bag-of-words model in which the feature values of each dimension are non-negative (e.g., term count or TF-IDF), guaranteeing the cosine between any two vectors to be also non-negative, the feature values in a fixed-length dense representation such as word2vec can be negative. This implies that the cosine between two vectors can also be negative.

5.1.5 Item Embedding

Instead of relying only on individual terms that can indicate whether an item is useful, we want to use all of the semantic information represented by the combination of all the terms present in that item. As we did in Section 5.1.4, we use the same fixed-length dense vector representation of terms. But instead of extracting the core vocabulary, we construct a "core vector" in the embedding space.

Let $\overrightarrow{t}$ be the unit vector of a term t in the embedding space. We compute $\overrightarrow{d}$, the unit vector of an unlabeled item d, as the average of the vectors of its terms:

$$\overrightarrow{d} = \sum_{t \in d} count(t,d)\overrightarrow{t} \Bigg/ \left\| \sum_{t \in d} count(t,d)\overrightarrow{t} \right\|_2$$

Similarly we define unit vectors $\overrightarrow{p}$ and $\overrightarrow{n}$, respectively, for each positive item (i.e., each item with the highest possible score: 1 for TREC-8; 3 for LiveQA), and for each negative item (i.e., each item with a score of 0 or, for LiveQA, with the minimum score available if all reference answers have a score greater than 0).

We then define the "core vector" $\overrightarrow{v_R}$ as the difference, in the embedding space, between $\overrightarrow{P}$, the average of all positive vectors $p \in R^+$, and $\overrightarrow{N}$, the average of all negative vectors $n \in R^-$:

$$\overrightarrow{v_R} = \overrightarrow{P} - \overrightarrow{N}$$

To compute a comparable "unlabeled vector" for the unlabeled item, we subtract the average $\overrightarrow{N}$ of negative vectors from the unlabeled vector $\overrightarrow{d}$. The cosine of the resulting vector and the core vector gives us a preliminary relevance estimate:

$$EsRel'_{IE}(d,R) = cosine\left(\overrightarrow{v_R}, \overrightarrow{d} - \overrightarrow{N}\right)$$

This preliminary relevance estimate comes with a caveat, though. It is unlikely that vectors for different items will have identical cosines with the reference. Thus, some differences will be claimed between unlabeled items—even the bad ones—when in reality no meaningful differences exist. Even worse, whenever a system returns a random item, it might be considered better than returning nothing. To mitigate this problem, we set a lower threshold below which we consider all of the unlabeled items to be equally bad (i.e., they get a score of 0), and an upper threshold above which we consider all of the unlabeled items to be equally good (i.e., they get a maximum allowed relevance score). We set the lower threshold as an aggregated value over the preliminary relevance estimates of negative reference items, and the upper threshold as an aggregated value over the preliminary relevance estimates of positive reference items. If the preliminary relevance estimates of negative reference items are separable from those of positive references items, then the corresponding aggregated values are their maximum and the minimum, respectively. Otherwise (e.g., there are outlier reference items or inconsistencies in the assessments), we use the average as an aggregating function for both thresholds.

$$EsRel_{IE}(d,R) = \begin{cases} 0 & \text{if } EsRel'_{IE}(d,R) < \underset{n \in R^-}{\text{max-or-mean}}\left(EsRel'_{IE}(n,R)\right) \\ 1 & \text{if } EsRel'_{IE}(d,R) > \underset{p \in R^+}{\text{min-or-mean}}\left(EsRel'_{IE}(p,R)\right) \\ EsRel'_{AE}(d,R) & \text{otherwise} \end{cases}$$

5.1.6 Augmented Diversity Sampling

All three of our baseline relevance estimation methods, and both of the word embedding techniques that we have proposed above, rely only on relevance judgments made by TREC assessors for items that had been proposed by some participating system. In TREC-8, some of the teams participated with manual runs, which involve some degree of human intervention. Several of those manual runs were substantially better than all of the automated systems. The TREC 2015 LiveQA track had participations only from automated systems. It seems reasonable to expect that additional diversity sampling would have been useful in that case. One possible source of diverse answers are answers that were later provided on the Yahoo! Answers platform that were subsequently indicated as good by the original asker of the question. Of course, some questions never attract a good answer, some good answers go unrewarded by the asker, and the askers may of course judge answer quality differently than a TREC assessor would. Nonetheless, the benefits of additional diversity may outweigh such sparsity and consistency concerns. We therefore enrich the reference answers used for item embedding with the answers we crawled (see Section 5.3 below). We use these "best answers" as positive references (i.e., we assign them a score of 3). We then apply the same procedure described in Section 5.1.5 above.

5.2 Evaluating Reusability

In this section we introduce our two novel approaches to characterizing the reusability of a test collection.

5.2.1 Single-Ablation with Regression

The usual way in which system ablation has been done is to determine the degree to which the original ranking of participating systems is preserved when we substitute the official evaluation results (e.g., MAP or NDCG) of the ranked list that they produce with evaluation results based on estimated relevance judgments. We ablate the participating systems, one at a time, and consider the remaining systems as providing reference items. Topics that have no good items are not useful for evaluating a non-participating system. Thus, for every ablated system, we ignore all topics that have no good reference items. We compute the average (over all eligible topics) estimated score for every ablated system. We also compute the average of the official scores over all topics (including the ones that were not eligible for ablation).

We characterize the preservation of the ranking using Pearson rank (ρ_r) correlation coefficient [47].[5] Unlike the original Pearson correlation coefficient (ρ) [153], ρ_r gives high weight to the items that have high official scores, and imposes more

[5]The dissertation's author motivated the problem and conducted the simulation-based experiments. Ning Gao, the lead author of the paper, defined the expression of ρ_r and provided the proofs of its properties.

penalty to swaps that occur near the head of the official ranked list. Contrary to other ranking-based correlation coefficients that fail to detect the change in the score differences between systems that maintained their rank (such as Kendal's τ [67], τ_{AP} [149] and τ_{GAp} [45]), ρ_r is sensitive to those differences. Because we are most interested in comparisons among the best systems, and especially those that have the largest (ground truth) differences, Pearson rank appears to be the most useful correlation coefficient, compared to other alternatives.

5.2.2 Paired-Ablation Evaluation

One limitation of the single-ablation approach is that the correlation coefficients are ignorant of the statistical significance of the differences between the scores, in both the official and the estimated ones. That is, we could be rewarding or penalizing the estimators for chance only. Another problem is that the set of topics over which average system estimated scores are computed is not necessarily the same. To see why this is the case, consider what happens in LiveQA when the only system that contributed the only item scored by the relevance judges as 3 is ablated. In such a case, relevance estimators that rely on having positive examples (as do every one other than the item length baseline) simply cannot estimate relevance. We thus cannot estimate relevance for that topic for that system. If this effect were random then the true mean was unchanged and only variance would increase, but because this will likely happen more often for good systems than for bad ones, this may tend to somewhat underestimate the quality of good systems.

We can mitigate these limitations by performing a series of t-tests for every pair of systems, using the official and estimated scores. Namely, we suggest that any two systems that we want to compare, based on estimated relevance, need to share the same set of eligible topics and reference items. For this reason, we ablate two systems at a time. For each pair of systems, and before doing any ablation, we compute their P@N using the official evaluation results (N is 1 for LiveQA, or 10 for TREC-8 Ad Hoc as we show in Sections 5.4) and we run a paired two-sided t-test over all topics. Let A and B be two systems. For a given significance level (in this chapter, $\alpha = 0.05$), we denote by $A = B$ the case were A and B are statistically indistinguishable, regardless of the sign of the difference in their means. We denote by $A \neq B$ the case in which there is a statistically significant difference in the means of the scores of A and B, which we can further specify as $A > B$ (i.e., the mean of the score of A is greater than that of B) or vice-versa.

Next, we ablate the pair of systems, and repeat the same process of significance test for the estimated scores, but, when necessary, ignoring topics for which no good item is present within the references. Then, we compare the outcome of these t-tests, yielding five cases:

- $A > B \rightarrow A > B$: this is the best case scenario, where both the significance of the test and the sign of the difference are maintained when we use the estimated scores instead of the official ones.
- $A > B \rightarrow B > A$: this is the worst case scenario, where the significance of the test is maintained and the sign of the difference is flipped.

- $A = B \rightarrow A = B$: we consider this to be a good outcome, as the indistinguishability of the difference is maintained.

- $A = B \rightarrow A \neq B$: this is potentially an erroneous case, as an apparently statistically significant difference shows up (regardless of the sign) when relevance estimation is used that was not present when the actual judgments were available.

- $A \neq B \rightarrow A = B$: this is a somewhat less troubling erroneous case in which what should have been a statistically significant difference, regardless of the sign, is not detectable when relevance estimation is used.

In the results we report in Section 5.3.4, we show all of these comparisons. When we wish to report a single-value measure for a particular score estimator, we do so as a ratio of the concordant comparisons (i.e., $A > B \rightarrow A > B$ and $A = B \rightarrow A = B$) to all available comparisons. We refer to this single-value measure as "consistency."

5.3 TREC 2015 LiveQA Evaluation

The main motivation of this work is to make the TREC LiveQA test collections reusable for researchers who seek to build better systems. We focus, in this section, on the first edition of the track.

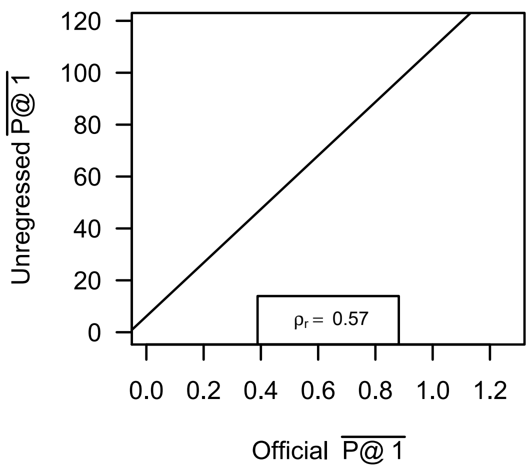
5.3.1 Test Collection

Twenty one systems participated in the TREC 2015 LiveQA track, returning 21,140 answers (all judged) for 1,087 questions. The official score reported for each system is the average, over all questions, of the answers returned by that system. In other words, it is the mean of the precision at 1, which we denote by $\overline{P@1}$. In addition to the official questions and answers introduced above, future answers to these questions that were posted on Yahoo! Answers were also crawled. We used only answers marked as "best answer" by the asker. A total of 176 questions did have such a best-answer. We optionally add these to our pool of answers, increasing the number of available perfect answers from 1,359 to 1,535 (i.e., +13%).

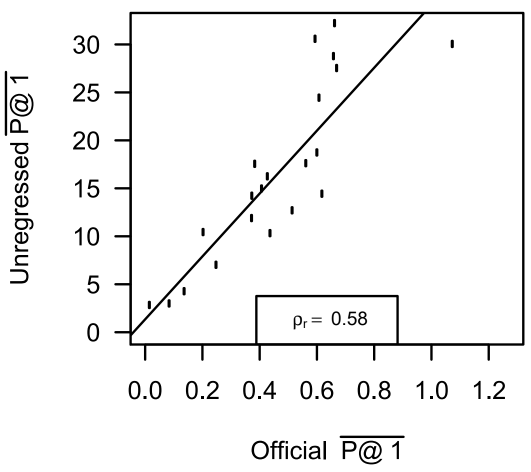
5.3.2 Single-Ablation Unregressed Results

Figure 5.1 depicts six scatter plots of the original versus the unregressed estimated $\overline{P@1}$, corresponding to the six relevance estimators. The ranges of the scores vary by estimator, and are far away from the range of the official scores. It is underestimated for the two item embedding methods, and overestimated for all of the other methods. Based on Pearson rank correlation coefficient, item length is, unsurprisingly, the worst relevance estimator. The best system is incorrectly outperformed by two systems (fourth and sixth dots from right to left), with a large margin in one case.

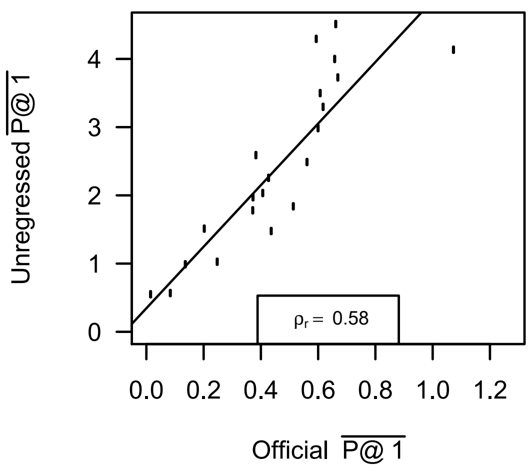
The clipped count and core vocabulary estimators are slightly better than the item length. Their main advantage is that the margin between the true best



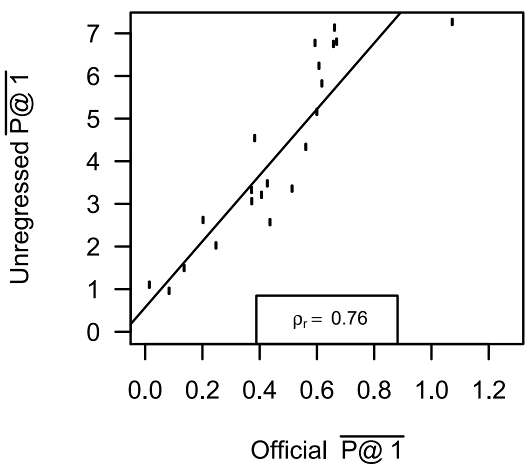
(a) Item length



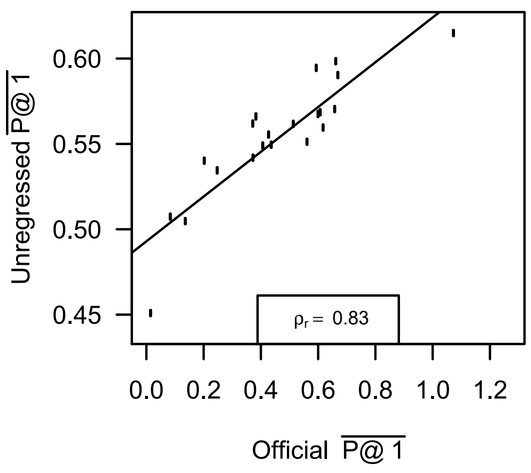
(b) Clipped count



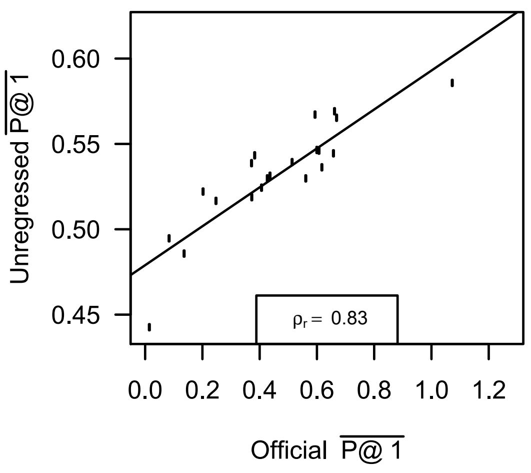
(c) Core vocabulary



(d) Expanded core vocabulary



(e) Item embedding



(f) Augmented item embedding

Figure 5.1: TREC-2015 LiveQA scores predicted by the unregressed relevance estimators in the single-ablation mode.

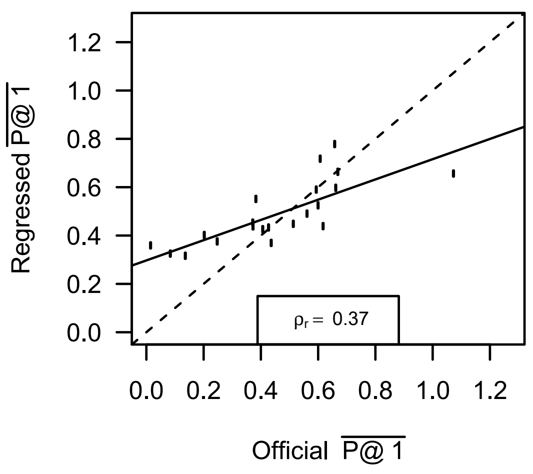
system and the incorrectly outperforming systems are narrower. Expanding the core vocabulary has a substantial impact (ρ_r increases from 0.58 to 0.76), as it preserves the ranking of the best system, but only at a small margin. It appears, thus, that the use of word embeddings helps reducing the lexical gap between the reference and ablated answers.

Item embedding performs better than all of the other relevance estimators, and seems to maintain better the relative gap between the systems, especially for the best one. But there is no evidence that augmenting item embedding is useful.

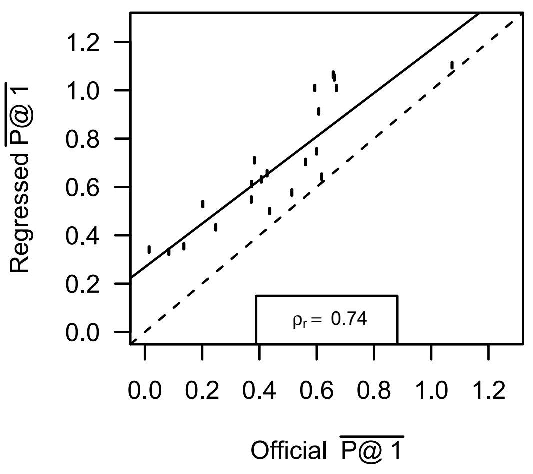
5.3.3 Single-Ablation Regressed Results

As shown in Section 5.3.2, the unbounded relevance estimators produce scores that are useful to rank systems and indicate how much one system is better than the other, but sometimes we need to compare absolute performance of a future system to the ones for which assessments are available. Figure 5.2 shows the scatter plots of the six relevance estimators when regression was performed on the unbounded estimated relevance scores.

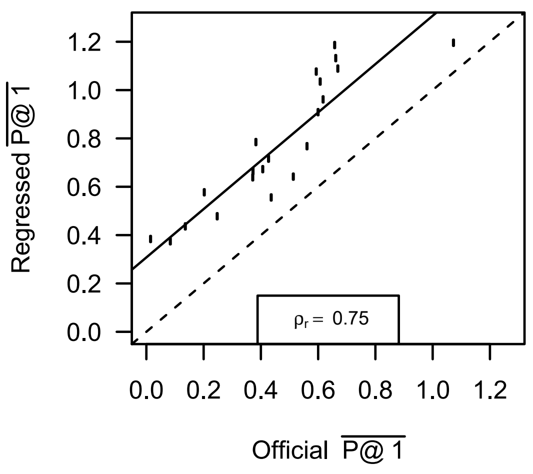
Except for item length, the regression has a benefit (when we compare against Figure 5.1) of improving the values of ρ_r, bringing them to 0.74 or higher. This is clearly because, in all of the five cases, the best performing system is correctly ranked first. Overall, it appears that applying the regression on relevance estimators with similar performance yields better results at the presence of more information from reference answers. Clipped count, which uses evidence of positive reference answers,



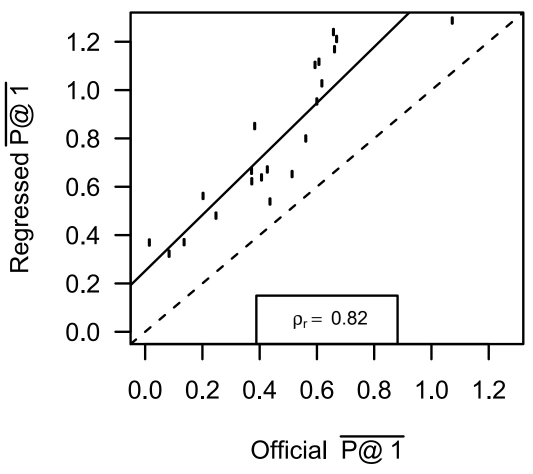
(a) Item length



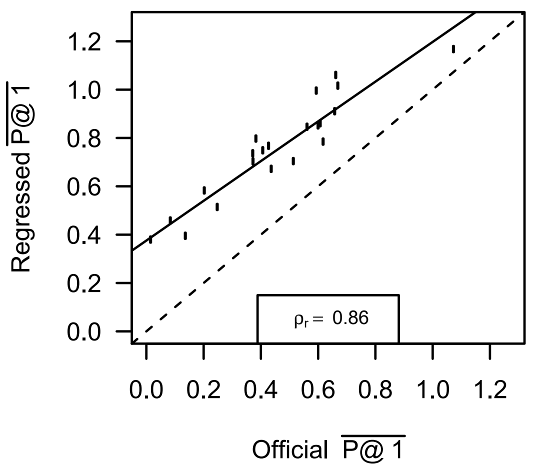
(b) Clipped count



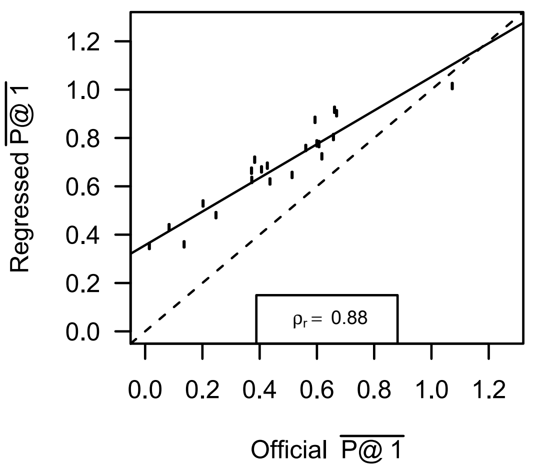
(c) Core vocabulary



(d) Expanded core vocabulary



(e) Item embedding



(f) Augmented item embedding

Figure 5.2: TREC-2015 LiveQA scores predicted by the regressed relevance estimators in the single-ablation mode.

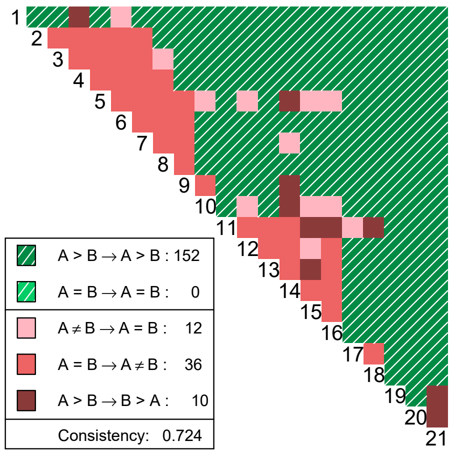
increases its ρ_r from 0.58 to 0.74, while item length, which uses no evidence from reference answers, decreases from 0.57 to 0.37. Similarly, core vocabulary, which uses evidence from both good and bad references improves (from 0.58 to 0.75) slightly better than clipped count (from 0.58 to 0.74), which uses only good references. Finally, augmented item embedding, which uses answers that were selected by the askers as best-answers, has its ρ_r increase from 0.83 to 0.88, which is higher than the increase observed for simple item embedding (from 0.83 to 0.86).

5.3.4 Paired-Ablation Results

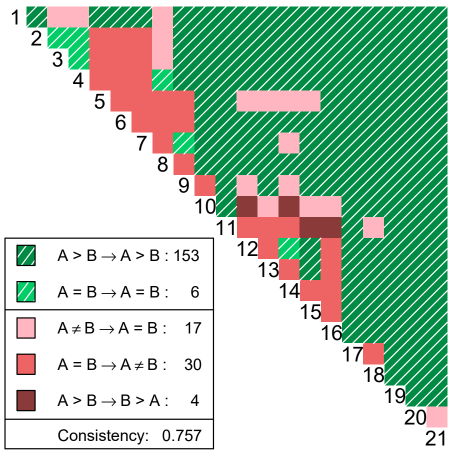
As discussed earlier in Section 5.2.2, a paired-ablation evaluation is more fair than single-ablation when we wish to compare two systems, because the underlying reference items are identical. In addition, it allows us to check whether the potential differences between these two systems would pass a statistical significance test.[6] Building on the results of Sections 5.3.2 and 5.3.3 that show that regression improves the performance of the relevance estimators, we limit ourselves in this subsection to paired-ablation with regression.

Figure 5.3 shows how the pairwise comparisons between each pair of systems changes when we substitute the official scores with the estimated scores. Answer length appears to have the worst consistency ($(152 + 0)/210 = 0.724$), followed

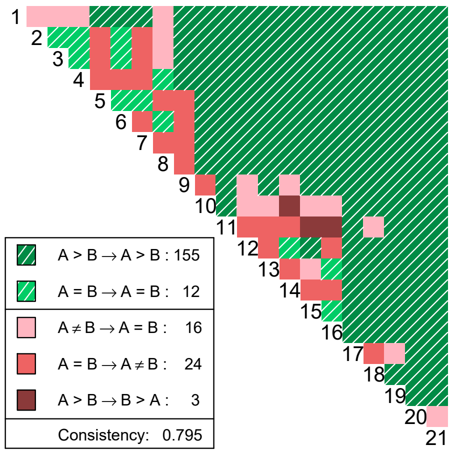
[6]Because we run many paired tests we don't claim that these tests as a group actually indicate significance; rather, our focus is on whether the results of any one test would change as the result of ablation (had that been the only test run).



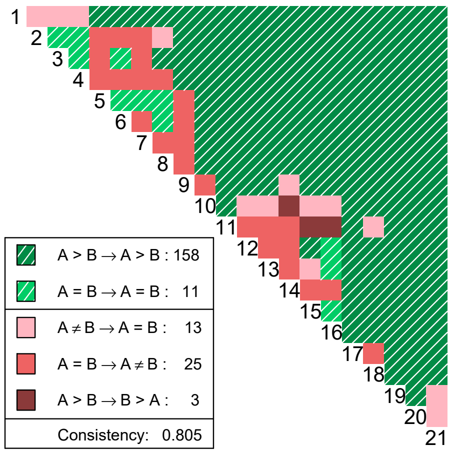
(a) Item length



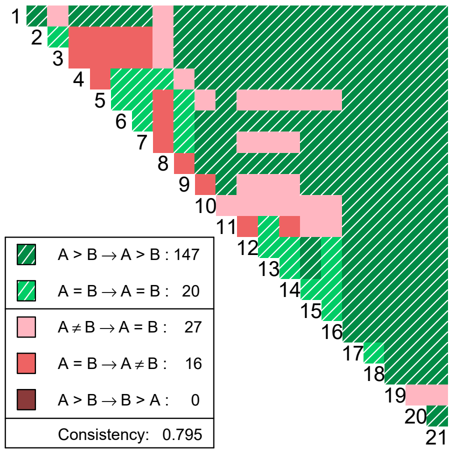
(b) Clipped count



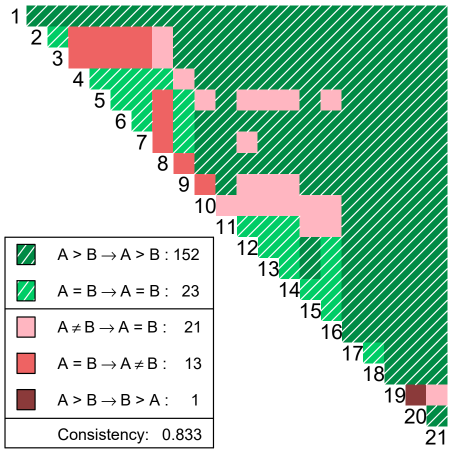
(c) Core vocabulary



(d) Expanded core vocabulary



(e) Item embedding



(f) Augmented item embedding

Figure 5.3: Maintenance of the (in)differences between TREC-2015 LiveQA systems. The systems are sorted in a decreasing order from left to right by the official scores.

by the clipped count $((153 + 6)/210 = 0.757)$. This observation stands out when we consider the most important error (i.e., $A > B \rightarrow B > A$), which occurs ten times for the former, and four times for the latter. Answer length has more tendency towards claiming significant differences $((152+36+10)/210 = 0.943)$, while in reality only $(152+12+10)/210 = 0.829$ of the differences are significant at the level of $\alpha = 0.05$. Core vocabulary beats the clipped count considering the overall consistency (i.e., 0.795 vs. 0.757), as well as each of its five components. Core vocabulary expansion improves the overall consistency slightly (from $(155+12)/210 = 0.795$ to $(158 + 11)/210 = 0.805$), as well as each of its five components.

While the overall consistency of simple item embedding is equal to that of core vocabulary, the former is better than all of the other relevance estimators when considering the important error—committing none. A conservative tendency is evident, underclaiming significant differences $((147+16)/210 = 0.776)$. Augmented item embedding has the highest overall consistency of $(152 + 23)/210 = 0.833$. The only important error it makes corresponds to a pair near the tail of the ranked list. Looking at the results from a different perspective, whenever this score estimator claims that a system is significantly better than another one, there is a chance of $152/(152 + 13) = 0.921$ that this is the same result that would have been seen with the unablated judgments. Perhaps the most important observation about augmented item embedding is that it is the only score estimator that correctly detects that the best system is significantly better than all other systems. This reinforces our observations from the single-ablation results (Section 5.3.3), in which this score estimator maintained a relatively large gap between the best and second-best systems. The sub-figure of augmented item embedding demonstrates, again, that enriching the references with diverse answers from Yahoo! Answers improves our ability to evaluate old ablated systems, and we would expect an even greater benefits from augmented item embedding for future (even better) systems.

5.4 TREC-8 Ad Hoc Evaluation

We have thus far focused our experiments on the TREC 2015 LiveQA test collection, as it exhibits the exact problem we want to solve: automated evaluation of future systems that return only documents that have not been retrieved before. With the exception of the evaluation campaigns oriented towards providing short answers (usually between a handful of words and a couple of sentences) to factoid questions, we are not aware of any test collection that has the same characteristics of the main issue we are addressing in this chapter. We leave factoid QA test collections for future work, and adapt a widely used test collection to our need.

The test collection created in the context of the TREC-8 Ad Hoc task [138] has been used extensively to develop algorithms and evaluation measures that can infer the relevance of unassessed documents based on their popularity [46, 128] or on their order within a ranked list containing documents of known relevance [19, 150]. In this track, 41 teams participated with a total of 129 systems, out of which 116 were fully automated and 13 involved some human-in-the-loop process (which TREC calls a "manual" run). They were asked to return up to 1,000 documents for each of 50 topics from a corpus of 528,155 documents. A pool of depth 100 was created from some selected systems. 4,728 out of 86,830 assessments were found to be relevant in total and MAP was reported as the official score for this task.

For the sake of visibility, we limit our experiments to single-ablation regressed evaluation (there are too many pairs to be shown on a paired-ablation figure). We perform the ablation in the following way. For any particular ablated system, we remove all of its documents, down to a depth N, from the pool of assessments.[7] We then estimate the relevance of these documents using all of the documents remaining in the pool. A choice of N that is very large will likely result in the disappearance of a large number of relevant documents from the pool, especially for good systems, thus penalizing the estimated score of these systems. A choice of N that is too small will increase the variance of our results, given the small number of 50 topics. Given that the average number of known relevant documents per topic is 94.56 in the TREC-8 Ad Hoc collection, we set N to 10—an arbitrary value that we expect is neither too large nor too small.

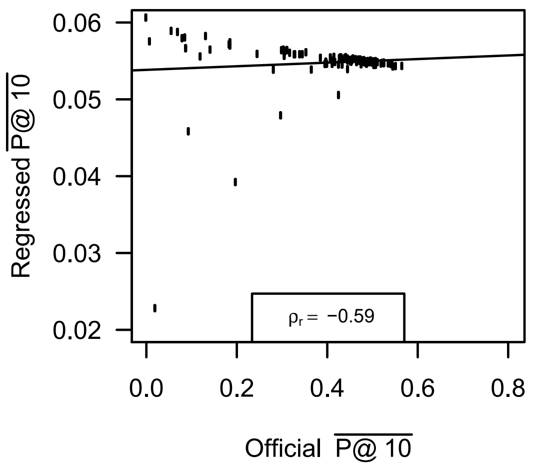
Recall that our regressed relevance estimators return a real number between 0 and 1. We opt, for simplicity, to report the scores based on the estimated precision at 10 (P@10) for both the official and estimated scores, which is the average over the scores over the top 10 documents.

[7]Note that we ablate not just the unique items by the ablated system (and no other system), but all the items found by that system (regardless of whether they were also found by other systems.) This is, thus, system-guided item ablation rather than system ablation. The motivation for this choice comes from the fact that our methods address only cases where none of the documents returned by a new system has ever been retrieved by any other old system.

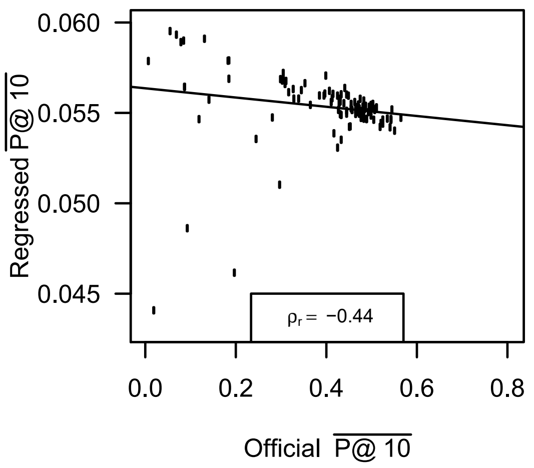
We use the same relevance estimators we experimented with for the TREC 2015 LiveQA track. By default, we use only the automatic systems. When the relevance estimator has, in addition, access to the manual run, we indicate that estimator to be an augmented one.

In Figure 5.4 we plot the scatter plots of the six relevance estimators. Item length, clipped count and core vocabulary (with and without expansion), all perform very poorly with negative ρ_r values between -0.59 and -0.43. This might be due to the sensitivity of these estimators to the variance of the item length. In fact, the answers of the TREC 2015 LiveQA track have 85 ± 63 terms on average, while this average is 343 ± 715 for the documents of the TREC-8 Ad Hoc collection.

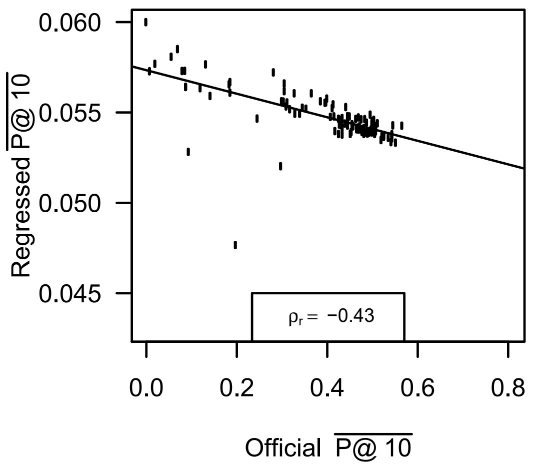
The item embedding relevance estimator, which is insensitive to the item length, performs relatively well in this test collection. It is slightly better with augmentation (ρ_r increases from 0.55 to 0.57). But, it fails to identify the best automatic system. However, when we include the the manual systems in the ablation study (i.e., we estimate their scores as well), we observe a considerable increase in Pearson rank to $\rho_r^{+o} = 0.83$. Several of these manual runs are substantially better than the automatic runs. This suggests that augmented item embedding succeeds in detecting best systems, particularly when their performance is largely better than the remaining systems.



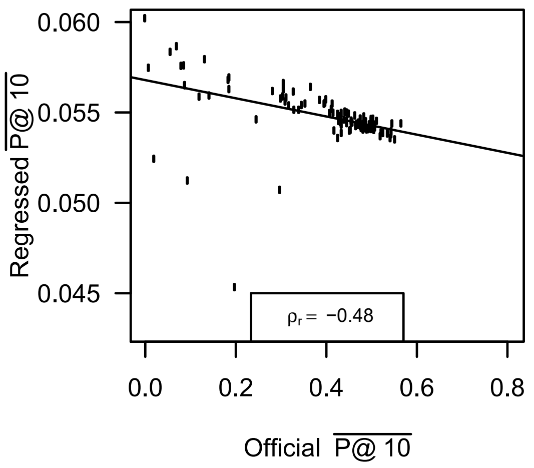
(a) Item length



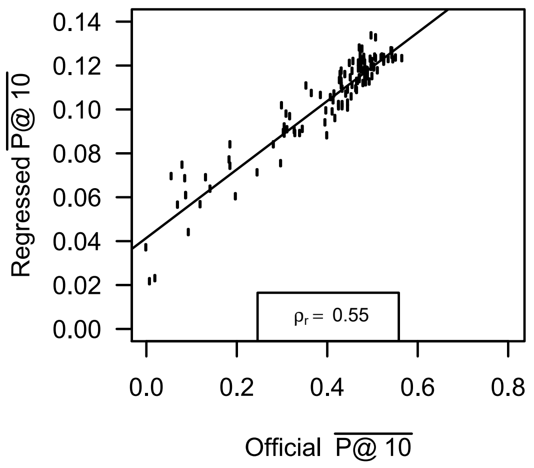
(b) Clipped count



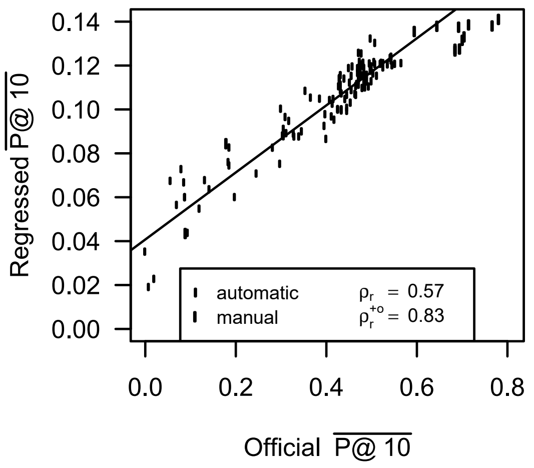
(c) Core vocabulary



(d) Expanded core vocabulary



(e) Item embedding



(f) Augmented item embedding

Figure 5.4: TREC-8 Ad Hoc scores predicted by the regressed relevance estimators in the single-ablation mode. The special Pearson rank value noted by ρ_r^{+o} corresponds to a case where we also ablate the manual runs.

5.5 Chapter Summary

In this chapter, we have introduced the problem of reusing test collections that are intended to be used to evaluate systems that retrieve only new documents. Based on three families of system ablations, we studied six relevance estimators, and found that one based on word embeddings and augmented with a diversity sampling approach provides a plausible estimate on the TREC 2015 LiveQA and TREC-8 Adhoc tracks. We have used this estimator to assess the performance of our Twitter-based answering system, for which we were missing relevance judgments (Section 4.4.3.2).

Chapter 6: Conclusion

Cross-platform question answering is a framework that aims to improve our ability to satisfy complex information needs by returning answers from different platforms, including those where the question has not been originally asked. This dissertation provides a proof of concept for this general idea (Contribution K1) by instantiating and studying some of its key components.

The input to a question answering system is the question itself. But questions are not always easily identifiable. Chapter 3 introduced the problem of detecting answerable questions in microblogging services, and suggested a pipeline for addressing it, pointing to the individual effectiveness of each of its main components (Contribution K4). We focused our experiments on two main filters: extracting qweets (i.e., questions that seek a real information need) out of all questions, and identifying aqweets (i.e., those that can be answerable by a stranger, and thus we would hope that a bot might be able to provide some useful answer). Our best classifier for detecting qweets is language agnostic and avoids the need for the hand-crafted feature engineering and selection that was performed in previous work (Contribution S2). This BLSTM neural model, when run with frozen trained embeddings, achieves an accuracy comparable to what Zhao and Mei report (compare our 0.851

to their 0.856), despite being trained on 21% less data. Its accuracy is considerably and significantly better than our reimplementation of that work (which has an accuracy of 0.805). It also yields the best results for Arabic, achieving substantially and statistically significantly better F_1 than Hasanain et al. (compare their 0.712 with our 0.784). A similar neural model performs well also on the novel problem of aqweet prediction (Contribution S3). Its superior precision of 0.674 is significantly higher than that of the best SVM model (0.464) in a 10-fold CV setup. We were able to remedy the low recall (in a fixed test-set setup), by labeling a limited number of tweets through active learning, increasing the recall from 0.330 to 0.535, while the annotation of a random sample of the same number of tweets would have resulted in a limited recall of 0.367. We are releasing our test collection for the task of detecting aqweets (5,000 tweets with random sampling and 2,000 tweets with active learning), the first of its kind to the best of our knowledge, to enable further research in this direction (Contribution C5).

After passing this filtering stage, the input question should next be handled by some answering platform. In Chapter 4, we looked at the problem of finding answers from different perspectives. We started by answering aqweets. We found that only 110 out of 362 answerable questions have at least one reply. With an average user satisfaction not exceeding 0.76/3 (which is attained after waiting for replies for more than one day), this suggests that another platform might fill in the information gap. We showed that a search for titles of old threads from Yahoo! Answers with content similar to that of the aqweets produces a great improvement in average user satisfaction (1.15/3 on a test set) compared to limiting the answers to the replies that the asker might have received on Twitter. While some techniques such as question rewriting did not demonstrate any statistically significant benefit, those and other simple relevance models contributed to an additional improvement based on a learning-to-rank model, attaining an average score of 1.36/3 on the test set. Our best model considers the content of the replies, as well as that of retrieved threads, before deciding to return an answer to the asker. By doing so, this classifier doubles the initial user satisfaction, reaching 1.59/3 (Contribution S6). We are releasing the annotations we have collected (4,881 relevance judgments distributed across 362 questions) to encourage more studies of aqweet answering (Contribution C7).

Next, we looked at the problem of answering complex questions such as the ones posted on CQA websites (Section 4.4). We examined two resources for answering questions asked on Yahoo! Answers: a crawl of old questions and answers from the same platform, and a large collection of tweets (Contributions K8 and S9). We showed that a system based on two scoring stages involving a deep neural network and a learning-to-rank model and that uses the Y!A crawl achieves, in expectation, the state-of-the-art and near-human-level effectiveness on automatic question answering, according to an independent evaluation within the TREC LiveQA track (Contribution S10). We also presented a system that searches in a filtered set of tweets and reranks the results using a learning-to-rank model trained on surrogate data from the TREC Microblog track (Contribution K11). This system demonstrated a substantial improvement in the expected retrieval effectiveness compared to a naive search of tweets. Our experiments also indicate that while the former system is generally a better choice, there are some types of questions, such as those about Travel, for which the two systems exhibit a comparable effectiveness (Contributions K12 and K13). This suggests that some combination of answers from the two platforms for those types of questions (as we did for answering aqweets) might lead to a better user satisfaction than if we would rely solely on either of them.

An important capability that a question answering system should implement is the ability to make a binary decision on returning answers (Sections 4.1.2 and 4.5). We addressed this problem first in the context of aqweet answering (Contribution S6). Given an answer returned by our best system from Yahoo! Answers, a good binary decision is to return it only when no replies to the original question are posted. Another method that delegates this decision to a SVM classifier has comparable effectiveness. We revisited this problem again in the context of the TREC RTS track, where we presented the architecture of a system that monitors a stream of tweets and makes a real-time decision for returning some of them to the user. This decision is based on a relevance-score threshold estimated automatically from old tweets. Our system has the highest precision among automatic systems that were evaluated by independent assessors on their mobile devices (Contribution S14).

Studying the usefulness of different platforms and building better answering systems require a substantial number of annotations. Considerable resources were allocated for generating and annotating the answers returned by several systems to over 2,000 questions in the two editions of the TREC LiveQA track. Reusing these annotations is, however, non-trivial. To make the annotations reusable, we introduced in Chapter 5 a new relevance estimator that uses labeled answers to assess the quality of unlabeled ones. Based on the embeddings of the terms of the labeled and unlabeled answers, this estimator works best when augmented with answers returned by humans (in addition to those selected by automatic systems), and when future systems (i.e., those for which we want to estimate the performance) are substantially better than the old ones (Contribution E16). We have used this estimator to evaluate our own Twitter-based system on the second edition of the TREC 2016 LiveQA track. We have also introduced three novel system-ablation approaches for evaluating the reusability of test collections (Contribution E15). Our single-ablation with regression approach is an improvement over typical traditional single-ablation, in which we map the estimated scores to the same score space of the original scores. We contributed to the definition of a new correlation measure, Pearson rank, which focuses on the maintenance of the differences between the scores of best ablated systems (Contribution E17), and which we adopted for our single-ablation study. The paired-ablation evaluation (with and without regression) provides a fair setup for comparing the maintenance of the direction and statistical significance of the differences between two ablated systems. We also introduced a new chart for visualizing these differences, and a single value, referred to as consistency, summarizing them.

6.1 Limitations

A number of important limitations should be kept in mind when interpreting the results reported in this dissertation. We note, in particular:

1. Cross-platform question answering can be implemented with tens of platform

types, and hundreds of instances. We limited ourselves to two instances (i.e., Twitter and Yahoo! Answers) from two platform types (i.e., microblogs and CQA services).

2. The experiments investigating answering aqweets (Section 4.3) were conducted on a set of tweets labeled to be aqweets. This is an optimistic setup, as in reality we would be implementing our pipeline of aqweet detection, which does not have a perfect accuracy. Hence, the user satisfaction expressed as an average score over the top-1 answer accuracy should be regarded as an upper bound of what we expect to achieve. Nevertheless, the comparison between different retrieval models is still fair, given that they all share the same set of true aqweets that was not intentionally selected to favor any particular model.

3. The size of the aqweet-answering collection (362 aqweets, and only one answer per retrieval model) limited our ability to detect statistical significance for some apparent differences. Similar experiments should be conducted using larger collections.

4. Both the aqweet detection (Section 3.3) and answering (Section 4.3) tasks might have suffered from some degree of experimenter bias. The researcher who developed the systems and analyzed the results has also defined the task, the annotation guidelines, and the test questions (on which the annotators were evaluated). In a better setup, we would prefer to decouple system development from data collection. Defining a task at some shared-task evaluation venue, such as TREC, might be a good next step in this respect.

5. User satisfaction, as measured by crowdsourcing workers or NIST assessors, is merely an approximation to the actual satisfaction by a third party. In reality, what might be a rhetorical question for someone can be a question seeking a real answer for somebody else. Similarly, two people receiving the same answer to an identical question might perceive it differently. We expect other factors of the answer utility (e.g., timeliness, source authority and diversity) to influence the actual perception of the questions and answers, but we decided to ignore such factors for the purpose of our study because doing so simplifies evaluation. Hence, the scores we have reported (i.e., in the effectiveness of the question detection and answering tasks) should be regarded as a basis for computing relative differences and ranking between the various methods for the aspect of system behavior that we have studied, rather than absolute numbers that fully characterize the user satisfaction.

6. The scores of our systems that participated in the TREC 2016 LiveQA track (Section 4.4.3), and on which we based some of our findings, are estimated scores (due to missing annotations). Our conclusions are subject to the assumptions that (1) the questions annotated for the CLIP-YA system are a representative sample of all questions of the track, and (2) the augmented item embeddings automatic performance estimator (applied to score CLIP-TW-A) has a good accuracy on that edition of the track.

7. Each of our TREC systems is an end-to-end live system structured as a pipeline of connected components (Sections 4.2, 4.4 and 4.5). While the evaluation

has demonstrated the good effectiveness of the overall systems, we are not able to draw a decisive conclusion about the influence of each component individually. Of course, now that the annotations have been released, we can try to isolate those components to examine which ones need more attention for improvement. Of course, any such effort will be subject to the limitations of evaluation with incomplete judgments.

8. The work on automatic evaluation of future answering systems (Chapter 5) might have been overfitted to the TREC 2015 LiveQA track. In fact, we have observed a lower correlation with actual results on the TREC-8 Ad Hoc experiments (although augmented item embedding was the best estimator in both cases), and we do not know whether the first edition of the LiveQA track was a special case, or whether our methods work only on question answering passage retrieval (for which the TREC-8 Ad Hoc track was not an instance). To answer this question, we suggest replicating the ablation studies on the TREC 2016 LiveQA track.

6.2 Lessons Learned

Throughout this dissertation, we have learned some lessons that might be useful for researchers and practitioners who want to continue the work on cross-platform question answering.

6.2.1 Big Data Win

Acquiring local access to a large repository of past questions and answers is beneficial to non-factoid question answering from different perspectives. First, we get control over where, when, and how to search. For instance, we can determine the fields to be searched, which is typically not possible in Web search. We can run the search several times at any moment, without worrying about connection latency or exceeding some Web search API quota. We can also try different tokenization/stemming choices and scoring functions, instead of being at the mercy of mysterious decisions made by a remote search engine that might be tuned to tasks other than ours. Second, we get our hands on large and free training data. This is particularly useful when existing test collections, such as those built in the TREC LiveQA track, are relatively small. The size of those small collections make them more suitable for tuning some parameters of the classifier, instead of training it. Of course, this big-data approach has its own challenges and limitations. The initial crawl might be prohibitive, and a distributed crawler is sometimes needed. That crawl might become outdated over time, calling for periodic or real-time refreshment. The hardware cost for storing the index and performing the search moves from the online search engine to the local search engine.

6.2.2 Fast Data Win

In near-real-time applications, speed is perhaps as important as effectiveness. With large indexed corpora, and long questions used as queries, disk input and output operations become lengthy on hard disk drives. As the price of solid-state drives is getting more affordable, using them for storing the inverted indexes should emerge as a natural solution.

6.2.3 Smart Data Win

While an enormous number of tweets are being publicly posted at any given time, only a small fraction of them are expected to have useful content for the questions we anticipate. Out of those, we can access only about 1% through Twitter's sample stream. That is, we are left with little useful content. To mitigate this limitation, we can leverage the knowledge we have about the categories of questions we expect to receive to track the tweets that contain the vocabulary related to any given category. By doing so, we get control over what is contained within the 1% set of tweets, increasing our chances of acquiring useful answers to potential questions.

6.2.4 Platform Agnostic Answering Pipeline

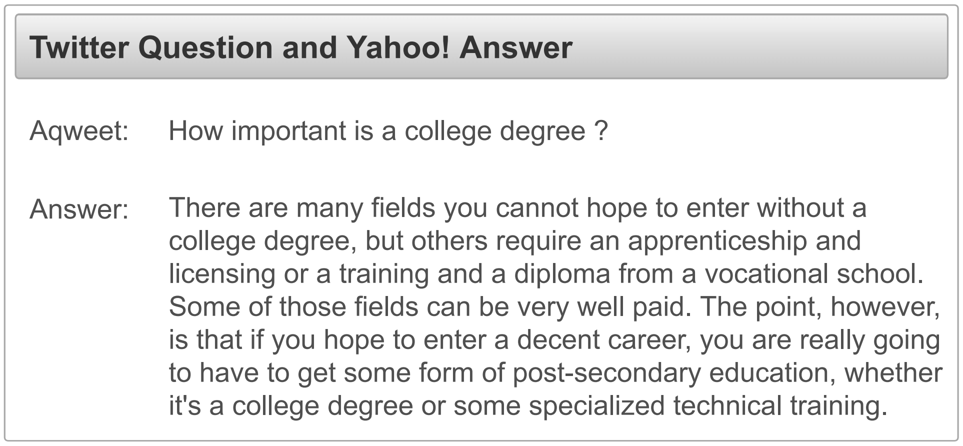
The pipeline we proposed for cross-platform question answering should be a default approach whenever we decide to integrate a new answering platform. After (1) selecting the answering platform, (2) a substantial amount of data needs to be collected. This collection is the basis for (3) a vanilla search with some state-of-the-art scoring function such as BM25 that aims to retrieve a list of candidate answers, and can also be used to (4) train and apply a re-ranking classifier that uses additional features that are specific to that answering platform.

6.2.5 One-Shot Crowdsourcing Annotations Are Safer

Incremental population of test collections, in which (1) we develop a system, (2) we collect annotations for its output, (3) we use those annotations to improve that system, and (4) we get additional annotations only for new documents that were found by the improved system, can help us to control the annotation budget while producing useful effectiveness estimates. With crowdsourcing, however, this might not necessarily be true. At every iteration, we get a different set of annotators who might annotate answers in a different manner compared to previous annotators. When this happens, it is difficult to attribute the difference we observe in the improved system to its actual effectiveness, or to the assessment behavior across different sets of annotators. To mitigate this limitation, we propose to gather a new set of assessments for all systems in a single shot, after we finish with all iterations.

6.2.6 BLSTM Classifiers are Often a Good Choice

Across the various classification problems we have studied in this dissertation, BLSTM seems to provide a good effectiveness, especially when there is a substantial amount of (free) training data. We suggest that this family of deep neural networks be tried first when approaching similar tasks (e.g., detection of information needs, query intent classification, and re-ranking of retrieved documents). On the other hand, some simple techniques work sometimes fairly well. For example, the Jaccard similarity measure was useful as a feature for deciding whether to return a candidate answer, and for detecting near-duplicate tweets. Such simple features should be considered when we have a limited amount of training data.


Twitter Question and Yahoo! Answer

Aqweet: How important is a college degree ?

Answer: There are many fields you cannot hope to enter without a college degree, but others require an apprenticeship and licensing or a training and a diploma from a vocational school. Some of those fields can be very well paid. The point, however, is that if you hope to enter a decent career, you are really going to have to get some form of post-secondary education, whether it's a college degree or some specialized technical training.

Figure 6.1: From a live demo for answering aqweets using Yahoo! Answers.

6.2.7 Task Specific Effectiveness Tuning

Different goals for a given problem might suggest using different values for the parameters of the model built to solve it. For instance, in the aqweet detection problem (Section 3.3), we were trying to balance between precision and recall to maximize the F_1 score. For aqweet answering, we were assuming that returning any answer for an aqweet that has no reply is better than returning nothing. However, for the sake of demonstrating our end-to-end aqweet answering system (i.e., from the publication of the tweet to returning the answer), we tuned our parameters to prefer high precision, allowing some sacrifice in recall, which is justified by the high volume of tweets. In an ad-hoc manner, we set a high threshold for deciding whether a candidate tweet is an aqweet, and we required a high similarity value (using the Quora-based classifier of Section 4.3.5) between the aqweet and the title of the candidate thread from Yahoo! Answers. As a result, the question / answer pairs that were detected by a real-time demonstration system and shown live during the defense of this dissertation were often nicely matched (e.g., Figure 6.1).

6.3 Future Work

Our experience suggests several directions for future work.

1. Our experiments in this dissertation were focused on one type of answering, which is searching for answers. More experiments need to be conducted on the complimentary approach of answering by crowdsourcing. Consider for instance a question that is too specific to be answered by returning an old answer. In that case, we can try to find a user who is an expert on the broad topic of the question, and who might be willing to provide an answer within certain desirable period. While there has been substantial work on detecting experts and routing questions to potential answerers (Section 2.6), doing this across different platforms is worth studying, as it increases the chances of satisfying the information need of the asker.

2. The evaluation of aqweet answering was inspired by the evaluation setup of the TREC LiveQA track, especially for the range of scores [0-3]. With such scoring, it is always better to return some answer instead of nothing. However, there is a crucial difference between answering aqweets and answering Y!A questions. In the latter case, we expect the asker to be looking for answers from strangers (although not necessarily a bot). But in the former case, we anticipate that the asker would be primarily calling for answers from her own friends and followers. Hence, our unsolicited answer might be unappreciated, especially if it turns out to be not useful. This can get even worse if the

predicted aqweet is a false positive (e.g., we answered something that isn't a question in the first place). We suggest three directions for addressing these concerns. First, a user study should quantify the degree of tolerance to an unsolicited answer, considering the quality of the answer and the surprise of the asker. Second, such quantification should be reflected in a new evaluation measure, perhaps one inspired by the expected gain measure of the TREC RTS track (adding a penalty when no answer should have been provided). Third, answering systems should be trained to optimize this new evaluation measure. In particular, they need to learn to make a binary decision on whether to return answers.

3. Breaking the aqweet detection task into three main stages (i.e., detection of itweets, qweets and then aqweets), was a natural approach when we were discovering the problem. Now that we have demonstrated that answerable questions can be detected, we can revisit those stages and merge some or all of them. In particular, we might benefit from the fact that all of the negative tweets at certain stage are also negative tweets in the subsequent stages. We might also take advantage of the fact that a similar neural architecture was designed for detecting both qweets and aqweets. Thus, we can for instance train a single classifier using the aqweets corpus, in addition to non-qweets from the corpus released by Zhao and Mei [156].

4. It is not clear how much credit should be attributed to the (lower) embedding layer versus the (higher) recurrent layer in our BLSTM networks that detect

qweets and aqweets. In other words, are the improvements over SVM due to the reduction of lexical mismatch between the terms? Or does the structure of the question make a contribution? To answer these questions, we suggest trying deep averaging networks [58], which ignore the order of the term sequence, considering a sentence to be a simple bag of vectors.

5. The paired-ablation evaluation is agnostic to the ranking of the systems and the differences in scores between them. Our consistency measure, for instance, gives equal penalties for swapping the top two systems or the bottom systems. A more desirable measure should penalize swapping the top pair more than the bottom pair, and we could use our work on Pearson rank to define a better consistency measure. For the visualization chart as well, we could make the hight of each row in a manner that is inversely proportional to the ranking of its corresponding system.

6. The two methods we introduced for system ablation (i.e., regressed single ablation, and regressed paired ablation) can be used for evaluating future (and better) score estimators. The single-value measure we suggested for each of the two ablation modes (i.e., Pearson rank and consistency) enables a direct comparison between different estimators.

7. Our score estimator can be used when the content of the documents of IR test collections cannot be made publicly available. Consider for instance the TREC Microblog and RTS tracks [84, 85], where the relevance judgments were released for tweet identifiers, but the contents of the tweets have to

be obtained directly from Twitter by the participants (because of Twitter's terms of service). The deletion of tweets over time (refer to our work on their prediction [8]), causes "swiss cheese" decay of the Microblog test collection. However, other tweets with similar content might be still available. If the track organizers release an embedded core vector for each annotated topic, the effort spent on the annotations can be exploited for a longer period by evaluating new systems retrieving similar tweets using those core vectors.

6.4 Implications

Some of the problems we have studied and the techniques we have developed have the potential for impact beyond the narrow scope of this dissertation.

6.4.1 Other Applications of Item Embedding

While item embedding was shown to be useful for estimating the effectiveness of systems that did not participate in the creation of the pool of documents that were assessed by human annotators, that technique might also be beneficial in other settings. One such case is where strong restrictions are enforced on the distribution of documents. For example, Twitter's terms of service prohibit the distribution of the content of tweets, and this is why the organizers of the TREC Microblog and Real-Time Summarization tracks distribute the identifiers of the tweets instead of their content. As tweets get deleted over time, newcomers who want to reproduce previous results or develop new techniques may suffer from the diminishing number of available annotations. Even those who already have access to those deleted tweets are required to delete their own copies. This simply means that the shelf life of such corpora are shorter than that of traditional test collections. However, if the organizers would release an embedding vector for the relevant tweets, and another one for non-relevant tweets, then the effort invested by the participating systems (to find relevant tweets) and the organizers (in the preparation of the topics and the annotations) could retain their value for a longer period. We would be able to assess the effectiveness of future systems using the old topics and embedding vectors and some future tweets. Of course, there would be some measurement error that needs to be characterized.

Another use case would be evaluation using new documents even when old test collections are still fully available. Consider for example a case in which we have limited resources to crawl one out of a candidate set of online travel forums. We could use the set of *Travel* questions from LiveQA that have some good answers, issue each of them as a query to some major Web search engine with a restriction to one candidate forum at a time, and then estimate the relevance of some top N snippets using the positive and negative embedding vectors for those questions. We might, for example, expect this approach to inform our decision about selecting which platforms to crawl first.

6.4.2 The Future of Question Answering

Research in information retrieval has for a long time been focusing on developing methods for satisfying information needs. But detecting those information needs is an area of research that has received less focus. For example, search engines assume that the query issued in an input box is a formulation of an information need, and the information that is sought in an evaluation campaign (such as in TREC) is typically given as a <query, description, narrative> triplet to search systems. However, focusing exclusively on such approaches would limit our ability to satisfy an information need to the cases where some statement of the information need is already at hand. Automatic identification of stated information needs appears to be a low-hanging fruit, for which the techniques we have explored in this dissertation might be applicable. In particular, looking at the broader implications of our work on detecting aqweets, we note that while the fraction of qweets that are aqweets is small, the amount of language produced by an average person on a daily basis (about 16,000 words [97]) far exceeds the content in the tweets posted in a day by an average user (about 5 tweets[1]). Hence, we might expect a larger impact from something like our aqweet detector if it were consolidated in a smart device, such as as a feature of some virtual assistant. Currently, these assistants are typically triggered by some specific keyword or phrase. A better interaction might sometimes result when a virtual assistant detects a question asked naturally (e.g., within a conversation between two individuals), and proactively provides some answers.

[1]https://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm

With more questions detected and higher expectations from the askers, one-size-fits-all answering systems of the type we have focused on in this dissertation will undoubtedly need to further evolve to address increasingly diverse and complex information needs. The emergence of specialized communities such as online forums for peer-based medical support, expatriate forums for people moving abroad, online tax and legal counseling services, and Q&A broadcasting sessions for clerical verdicts, provide a glimpse on what the panorama of answering platforms will look like in the future. Some of those websites have their content dominated by some specific languages. We anticipate that accuracy, timeliness, personalization, privacy, cross-lingual access will all be competing factors for increasingly sophisticated algorithms for cross-platform question answering. This dissertation is just the beginning.

Bibliography

[1] Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna Harman. Overview of the TREC 2015 LiveQA track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/Overview-QA.pdf.

[2] Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna Harman. Overview of the TREC 2016 LiveQA track. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. Notebook version.

[3] Weijie An, Mengfei Shi, Xin Ouyang, Yan Yang, Qinmin Hu, and Liang He. ECNU at 2016 LiveQA track: A parameter sharing long short term memory model for learning question similarity. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/ECNU-QA.pdf.

[4] Jaime Arguello. Aggregated search. Foundations and Trends® in Information Retrieval, 10(5):365–502, 2017. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000052.

[5] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. Sources of evidence for vertical selection. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, pages 315–322, Boston, MA, USA, 2009. ACM. ISBN 978-1-60558-483-6. URL http://doi.acm.org/10.1145/1571941.1571997.

[6] Mossaab Bagdouri and Douglas W. Oard. CLIP at TREC 2015: Microblog and LiveQA. In Proceedings of the Twenty-Fourth Text REtrieval Conference, TREC, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/CLIP-MBQA.pdf.

[7] Mossaab Bagdouri and Douglas W. Oard. Profession-based person search in microblogs: Using seed sets to find journalists. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, pages 593–602, Melbourne, Australia, 2015. ACM. ISBN 978-1-4503-3794-6. URL http://doi.acm.org/10.1145/2806416.2806466.

[8] Mossaab Bagdouri and Douglas W. Oard. On predicting deletions of microblog posts. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, pages 1707–1710, Melbourne, Australia, 2015. ACM. ISBN 978-1-4503-3794-6. URL http://doi.acm.org/10.1145/2806416.2806600.

[9] Mossaab Bagdouri and Douglas W. Oard. CLIP at TREC 2016: LiveQA and RTS. In Proceedings of the Twenty-Fifth Text REtrieval Conference, TREC, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/CLIP-QA-RT.pdf.

[10] Mossaab Bagdouri and Douglas W. Oard. Building bridges across social platforms: Answering Twitter questions with Yahoo! Answers. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, Tokyo, Japan, 2017. ACM. ISBN 978-1-4503-5022-8. URL http://doi.acm.org/10.1145/3077136.3080755.

[11] Mossaab Bagdouri and Douglas W. Oard. Detecting answerable questions in microblogs. In preparation.

[12] Mossaab Bagdouri and Douglas W. Oard. On the reusability of open-resource test collections: Estimating relevance with word embeddings. Under review.

[13] Ziv Bar-Yossef and Maxim Gurevich. Random sampling from a search engine's index. In Proceedings of the 15th International Conference on World Wide Web, WWW '06, pages 367–376, Edinburgh, Scotland, 2006. ACM. ISBN 1-59593-323-9. URL http://doi.acm.org/10.1145/1135777.1135833.

[14] Parantapa Bhattacharya and Niloy Ganguly. Characterizing deleted tweets and their authors. In Proceedings of the 10th International AAAI Conference on Weblogs and Social Media, ICWSM '16, Cologne, Germany, 2016. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM16/paper/view/13133.

[15] Dasha Bogdanova, Debasis Ganguly, Jennifer Foster, and Ali Hosseinzadeh Vahid. ADAPT.DCU at TREC LiveQA: A sentence retrieval based approach to live question answering. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/ADAPT.DCU-QA.pdf.

[16] Mohamed Bouguessa, Shengrui Wang, and Benoit Dumoulin. Discovering knowledge-sharing communities in question-answering forums. ACM Transactions on Knowledge Discovery from Data, 5(1):3:1–3:49, December 2010. ISSN 1556-4681. URL http://doi.acm.org/10.1145/1870096.1870099.

[17] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference.

In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP '15, Lisbon, Portugal, 2015.

[18] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '00, pages 33–40, Athens, Greece, 2000. ACM. ISBN 1-58113-226-3. URL http://doi.acm.org/10.1145/345508.345543.

[19] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '04, pages 25–32, Sheffield, United Kingdom, 2004. ACM. ISBN 1-58113-881-4. URL http://doi.acm.org/10.1145/1008992.1009000.

[20] Stefan Büttcher, Charles L. A. Clarke, and Ian Soboroff. Overview of the TREC 2006 terabyte track. In Proceedings of the Fifteenth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2006. URL http://trec.nist.gov/pubs/trec15/papers/TERA06.OVERVIEW.pdf.

[21] Stefan Büttcher, Charles L. A. Clarke, Peter C. K. Yeung, and Ian Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, pages 63–70, Amsterdam, The Netherlands, 2007. ACM. ISBN 978-1-59593-597-7. URL http://doi.acm.org/10.1145/1277741.1277755.

[22] Jamie Callan. Distributed information retrieval. In Advances in Information Retrieval, volume 7 of The Information Retrieval Series, pages 127–150. Springer US, 2000. ISBN 978-0-7923-7812-9. URL http://dx.doi.org/10.1007/0-306-47019-5_5.

[23] Jamie Callan and Margaret Connell. Query-based sampling of text databases. ACM Transactions on Information Systems, 19(2):97–130, April 2001. ISSN 1046-8188. URL http://doi.acm.org/10.1145/382979.383040.

[24] Ben Carterette and James Allan. Semiautomatic evaluation of retrieval systems using document similarities. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pages 873–876, Lisbon, Portugal, 2007. ACM. ISBN 978-1-59593-803-9. URL http://doi.acm.org/10.1145/1321440.1321564.

[25] Ben Carterette, James Allan, and Ramesh Sitaraman. Minimal test collections for retrieval evaluation. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, pages 268–275, Seattle, Washington, USA, 2006. ACM. ISBN 1-59593-369-7. URL http://doi.acm.org/10.1145/1148170.1148219.

[26] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(3): 27:1–27:27, May 2011. ISSN 2157-6904. URL http://doi.acm.org/10.1145/1961189.1961199.

[27] Shuo Chang and Aditya Pal. Routing questions for collaborative answering in community question answering. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '13, pages 494–501. ACM, 2013. ISBN 978-1-4503-2240-9. URL http://doi.acm.org/10.1145/2492517.2492559.

[28] Ruey-Cheng Chen, J. Shane Culpepper, Tadele Tadela Damessie, Timothy Jones, Ahmed Mourad, Kevin Ong, Falk Scholer, and Evi Yulianti. RMIT at the TREC 2015 LiveQA track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/RMIT-QA.pdf.

[29] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In DLRL Workshop, 2014.

[30] Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam, and Egidio L. Terra. Question answering by passage selection. In Tomek Strzalkowski and Sanda M. Harabagiu, editors, Advances in Open Domain Question Answering, volume 32 of Text, Speech and Language Technology, pages 259–283. Springer Netherlands, 2006. ISBN 978-1-4020-4744-2. URL http://dx.doi.org/10.1007/978-1-4020-4746-6_8.

[31] Gordon V. Cormack, Maura R. Grossman, Bruce Hedin, and Douglas W. Oard. Overview of the TREC 2010 legal track. In Proceedings of the Nineteenth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2010. URL http://trec.nist.gov/pubs/trec19/papers/LEGAL10.OVERVIEW.pdf.

[32] Josue Balandrano Coronel. University of Texas Rio Grande Valley TREC LiveQA 2016: Using topic modeling to answer complex questions. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/JBC-TREC2016-QA.pdf.

[33] Kareem Darwish and Walid Magdy. Arabic information retrieval. Foundations and Trends® in Information Retrieval, 7(4):239–342, 2014. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000031.

[34] Kareem Darwish and Douglas Oard. Probabilistic structured query methods. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '03, pages 338–344, Toronto, Canada, 2003. ISBN 1-58113-646-3. URL http://doi.acm.org/10.1145/860435.860497.

[35] Kareem Darwish, Walid Magdy, and Ahmed Mourad. Language processing for Arabic microblog retrieval. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, pages 2427–2430, Maui, Hawaii, USA, 2012. ISBN 978-1-4503-1156-4. URL http://dx.doi.org/10.1145/2396761.2398658.

[36] Vivek Datla, Sadid A. Hasan, Joey Liu, Yassine Benajiba, Kathy Lee, Ashequl Qadir, Aaditya Prakash, and Oladimeji Farri. Open domain real-time question answering based on semantic and syntactic question similarity. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/prna-QA.pdf.

[37] Gideon Dror, Yehuda Koren, Yoelle Maarek, and Idan Szpektor. I want to answer; who has a question?: Yahoo! Answers recommender system. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, pages 1109–1117. ACM, 2011. ISBN 978-1-4503-0813-7. URL http://doi.acm.org/10.1145/2020408.2020582.

[38] Gideon Dror, Yoelle Maarek, and Idan Szpektor. Will my question be answered? Predicting "question answerability" in community question-answering sites. In Machine Learning and Knowledge Discovery in Databases, volume 8190 of Lecture Notes in Computer Science, pages 499–514. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40993-6. URL http://dx.doi.org/10.1007/978-3-642-40994-3_32.

[39] Olive Jean Dunn. Multiple comparisons among means. Journal of the American Statistical Association, 56(293):52–64, 1961. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1961.10482090.

[40] Miles Efron and Megan Winget. Questions are content: A taxonomy of questions in a microblogging environment. Proceedings of the American Society for Information Science and Technology, 47(1):1–10, 2010. ISSN 1550-8390. URL http://dx.doi.org/10.1002/meet.14504701208.

[41] Matthew Ekstrand-Abueg, Virgil Pavlu, and Javed A. Aslam. Live nuggets extractor: A semi-automated system for text extraction and test collection creation. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, pages 1087–1088, Dublin, Ireland, 2013. ACM. ISBN 978-1-4503-2034-4. URL http://doi.acm.org/10.1145/2484028.2484211.

[42] George Forman. An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research, 3:1289–1305, March 2003. ISSN 1532-4435.

[43] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5):1189–1232, 10 2001. URL http://dx.doi.org/10.1214/aos/1013203451.

[44] Gabriel Pui Cheong Fung, Jeffrey X. Yu, Hongjun Lu, and Philip S. Yu. Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering, 18(1):6–20, January 2006. ISSN 1041-4347. URL http://dx.doi.org/10.1109/TKDE.2006.16.

[45] Ning Gao and Douglas Oard. A head-weighted gap-sensitive correlation coefficient. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pages 799–802, Santiago, Chile, 2015. ACM. ISBN 978-1-4503-3621-5. URL http://doi.acm.org/10.1145/2766462.2767793.

[46] Ning Gao, William Webber, and Douglas W. Oard. Reducing reliance on relevance judgments for system comparison by using expectation-maximization. In Advances in Information Retrieval, volume 8416 of Lecture Notes in Computer Science, pages 1–12. Springer International Publishing, 2014. ISBN 978-3-319-06028-6. URL http://dx.doi.org/10.1007/978-3-319-06028-6_1.

[47] Ning Gao, Mossaab Bagdouri, and Douglas W. Oard. Pearson rank: A head-weighted gap-sensitive score-based correlation coefficient. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, pages 941–944. ACM, 2016. ISBN 978-1-4503-4069-4. URL http://doi.acm.org/10.1145/2911451.2914728.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. URL http://www.deeplearningbook.org.

[49] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of GIOSS for the text database discovery problem. In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, SIGMOD '94, pages 126–137, Minneapolis, Minnesota, USA, 1994. ACM. ISBN 0-89791-639-5. URL http://doi.acm.org/10.1145/191839.191869.

[50] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. Starts: Stanford proposal for internet meta-searching. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD '97, pages 207–218, Tucson, Arizona, USA, 1997. ACM. ISBN 0-89791-911-4. URL http://doi.acm.org/10.1145/253260.253299.

[51] Laszlo Grunfeld and Kui-Lam Kwok. Sentence ranking using keywords and meta-keywords. In Advances in Open Domain Question Answering, volume 32 of Text, Speech and Language Technology, pages 229–258. Springer Netherlands, 2006. ISBN 978-1-4020-4744-2. URL http://dx.doi.org/10.1007/978-1-4020-4746-6_7.

[52] Kilem L Gwet. Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters. Advanced Analytics, LLC, 2014.

[53] Sanda M. Harabagiu. Questions and intentions. In Advances in Open Domain Question Answering, volume 32 of Text, Speech and Language Technology, pages 99–147. Springer Netherlands, 2006. ISBN 978-1-4020-4744-2. URL http://dx.doi.org/10.1007/978-1-4020-4746-6_4.

[54] Maram Hasanain, Tamer Elsayed, and Walid Magdy. Identification of answer-seeking questions in Arabic microblogs. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, pages 1839–1842, Shanghai, China, 2014. ACM. ISBN 978-1-4503-2598-1. URL http://doi.acm.org/10.1145/2661829.2661959.

[55] Maram Hasanain, Mossaab Bagdouri, Tamer Elsayed, and Douglas Oard. What questions do journalists ask on Twitter? In Proceedings of the ICWSM Workshop on Social Media in the Newsroom, 2016. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM16/paper/view/13221.

[56] Katja Hofmann, Lihong Li, and Filip Radlinski. Online evaluation for information retrieval. Foundations and Trends® in Information Retrieval, 10(1):1–117, 2016. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000051.

[57] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP '14, pages 633–644, Doha, Qatar, 2014.

[58] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In Proceedings of the Association for Computational Linguistics, ACL '15, 2015.

[59] Jin-Woo Jeong, Meredith Morris, Jaime Teevan, and Dan Liebling. A crowd-powered socially embedded search engine. In Proceedings of the 7th International AAAI Conference on Weblogs and Social Media, ICWSM '13, 2013. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/5986.

[60] Thorsten Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, 1999. ISBN 0-262-19416-3.

[61] Thorsten Joachims. A support vector method for multivariate performance measures. In Proceedings of the 22nd International Conference on Machine Learning, ICML '05, pages 377–384, Bonn, Germany, 2005. ACM. ISBN 1-59593-180-5. URL http://doi.acm.org/10.1145/1102351.1102399.

[62] Thorsten Joachims. Training linear SVMs in linear time. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 217–226, Philadelphia, PA, USA, 2006. ACM. ISBN 1-59593-339-5. URL http://doi.acm.org/10.1145/1150402.1150429.

[63] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments. Information Processing and Management, 36(6):779–808, November 2000. ISSN 0306-4573. URL http://dx.doi.org/10.1016/S0306-4573(00)00015-7.

[64] Timothy Jones, Andrew Turpin, Stefano Mizzaro, Falk Scholer, and Mark Sanderson. Size and source matter: Understanding inconsistencies in test collection-based evaluation. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14, pages 1843–1846, Shanghai, China, 2014. ACM. ISBN 978-1-4503-2598-1. URL http://doi.acm.org/10.1145/2661829.2661945.

[65] Daniel Jurafsky and James H. Martin. Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 0131873210.

[66] Diane Kelly and Jimmy Lin. Overview of the TREC 2006 ciQA task. SIGIR Forum, 41(1):107–116, June 2007. ISSN 0163-5840. URL http://doi.acm.org/10.1145/1273221.1273231.

[67] M. G. Kendall. A new measure of rank correlation. Biometrika, 30(1-2):81–93, 1938. URL http://dx.doi.org/10.1093/biomet/30.1-2.81.

[68] Maria Khvalchik and Anagha Kulkarni. San Francisco State University at LiveQA track of TREC 2016. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/IR.SFSU.2016-QA.pdf.

[69] Yoon Kim. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP '14, Doha, Qatar, 2014.

[70] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In 3rd International Conference on Learning Representations, ICLR '15, San Diego, CA, USA, 2015.

[71] Paul Kingsbury and Martha Palmer. From treebank to propbank. In Proceedings of the Third International Conference on Language Resources and Evaluation, LREC '02, pages 1989–1993, Las Palmas, Canary Islands - Spain, 2002. European Language Resources Association (ELRA).

[72] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI '95, pages 1137–1143, Montreal, Quebec, Canada, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8.

[73] Robert Krovetz. Viewing morphology as an inference process. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93, pages 191–202, Pittsburgh, Pennsylvania, USA, 1993. ACM. ISBN 0-89791-605-0. URL http://doi.acm.org/10.1145/160688.160718.

[74] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. Biometrics, 33(1):159–174, 1977. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2529310.

[75] Matthew Lease and Emine Yilmaz. Crowdsourcing for information retrieval. SIGIR Forum, 45(2):66–75, January 2012. ISSN 0163-5840. URL http://doi.acm.org/10.1145/2093346.2093356.

[76] David D. Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. SIGIR Forum, 29(2):13–19, September 1995. ISSN 0163-5840. URL http://doi.acm.org/10.1145/219587.219592.

[77] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94, pages 3–12, Dublin, Ireland, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. URL http://dl.acm.org/citation.cfm?id=188490.188495.

[78] Baichuan Li and Irwin King. Routing questions to appropriate answerers in community question answering services. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pages 1585–1588, Toronto, Ontario, Canada, 2010. ACM. ISBN 978-1-4503-0099-5. URL http://doi.acm.org/10.1145/1871437.1871678.

[79] Baichuan Li, Xiance Si, Michael R. Lyu, Irwin King, and Edward Y. Chang. Question identification on Twitter. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, pages 2477–2480, Glasgow, Scotland, UK, 2011. ACM. ISBN 978-1-4503-0717-8. URL http://doi.acm.org/10.1145/2063576.2063996.

[80] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In Proceedings of the ACL Workshop on Text Summarization Branches Out, pages 74–81, Barcelona, Spain, 2004. ACL.

[81] Jimmy Lin and Dina Demner-Fushman. Automatically evaluating answers to definition questions. In Proceedings of the Conference on Human Language

Technology and Empirical Methods in Natural Language Processing, HLT '05, pages 931–938, Vancouver, British Columbia, Canada, 2005. ACL. URL http://dx.doi.org/10.3115/1220575.1220692.

[82] Jimmy Lin and Dina Demner-Fushman. Methods for automatically evaluating answers to complex questions. Information Retrieval, 9(5):565–587, 2006. ISSN 1386-4564. URL http://dx.doi.org/10.1007/s10791-006-9003-7.

[83] Jimmy Lin and Pengyi Zhang. Deconstructing nuggets: The stability and reliability of complex question answering evaluation. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, pages 327–334, Amsterdam, The Netherlands, 2007. ACM. ISBN 978-1-59593-597-7. URL http://doi.acm.org/10.1145/1277741.1277799.

[84] Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, and Ellen Voorhees. Overview of the TREC-2015 microblog track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/Overview-MB.pdf.

[85] Jimmy Lin, Adam Roegiest, Luchen Tan, Richard McCreadie, Ellen Voorhees, and Fernando Diaz. Overview of the TREC 2016 real-time summarization track. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/Overview-RT.pdf.

[86] Qiaoling Liu, Eugene Agichtein, Gideon Dror, Evgeniy Gabrilovich, Yoelle Maarek, Dan Pelleg, and Idan Szpektor. Predicting web searcher satisfaction with existing community-based answers. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, pages 415–424, Beijing, China, 2011. ACM. ISBN 978-1-4503-0757-4. URL http://doi.acm.org/10.1145/2009916.2009974.

[87] Tie-Yan Liu. Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval, 3(3):225–331, 2009. URL http://dx.doi.org/10.1561/1500000016.

[88] Zhe Liu and Bernard J. Jansen. Almighty Twitter, what are people asking for? Proceedings of the American Society for Information Science and Technology, 49(1):1–10, 2012. ISSN 1550-8390. URL http://dx.doi.org/10.1002/meet.14504901134.

[89] Zhe Liu and Bernard J. Jansen. Subjective versus Objective Questions: Perception of Question Subjectivity in Social Q&A, pages 131–140. Springer International Publishing, Cham, 2015. ISBN 978-3-319-16268-3. URL http://dx.doi.org/10.1007/978-3-319-16268-3_14.

[90] Marco Lui and Timothy Baldwin. Accurate language identification of Twitter messages. In Proceedings of the 5th Workshop on Language Analysis for Social Media @ EACL, LASM '14, pages 17–25, Gothenburg, Sweden, 2014.

[91] Mihai Lupu and Allan Hanbury. Patent retrieval. Foundations and Trends® in Information Retrieval, 7(1):1–97, 2013. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000027.

[92] Joel Mackenzie, Ruey-Cheng Chen, and J. Shane Culpepper. RMIT at the TREC 2016 LiveQA track. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/RMIT-QA.pdf.

[93] Walid Magdy, Wei Gao, Tarek Elganainy, and Zhongyu Wei. QCRI at TREC 2014: Applying the KISS principle for the TTG task in the Microblog track. In Proceedings of the Twenty-Third Text REtrieval Conference, 2014. URL http://trec.nist.gov/pubs/trec23/papers/pro-QCRI_microblog.pdf.

[94] Rana Malhas, Marwan Torki, Rahma Ali Evi Yulianti, and Tamer Elsayed. Real, live, and concise: Answering open-domain questions with word embedding and summarization. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/QU-QA.pdf.

[95] Gregory Marton and Alexey Radul. Nuggeteer: Automatic nugget-based evaluation using descriptions and judgements. In Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL '06, pages 375–382, New York, New York, 2006. ACL. URL http://dx.doi.org/10.3115/1220835.1220883.

[96] Edgard Marx and Sandro Coelho. Answering live questions from heterogeneous data sources: SMART in Live QA at TREC 2016. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/AKSW-QA.pdf.

[97] Matthias R. Mehl, Simine Vazire, Nairán Ramírez-Esparza, Richard B. Slatcher, and James W. Pennebaker. Are women really more talkative than men? Science, 317(5834):82–82, 2007. URL http://science.sciencemag.org/content/317/5834/82.

[98] Weiyi Meng, Clement Yu, and M. Tamer Ozsu. Advanced Metasearch Engine Technology. Morgan & Claypool Publishers, 2010. ISBN 1608451925, 9781608451920.

[99] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Proceedings of Workshop at

ICLR, Scottsdale, Arizona, USA, 2013. URL http://arxiv.org/abs/1301.3781.

[100] George A. Miller. WordNet: A lexical database for English. Communications of the ACM, 38(11):39–41, November 1995. ISSN 0001-0782. URL http://doi.acm.org/10.1145/219717.219748.

[101] Gilad Mishne, David Carmel, Ron Hoory, Alexey Roytman, and Aya Soffer. Automatic analysis of call-center conversations. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05, pages 453–459, Bremen, Germany, 2005. ACM. ISBN 1-59593-140-6. URL http://doi.acm.org/10.1145/1099554.1099684.

[102] Alistair Moffat, Falk Scholer, and Paul Thomas. Models and metrics: IR evaluation as a user process. In Proceedings of the Seventeenth Australasian Document Computing Symposium, ADCS '12, pages 47–54, Dunedin, New Zealand, 2012. ACM. ISBN 978-1-4503-1411-4. URL http://doi.acm.org/10.1145/2407085.2407092.

[103] Meredith Ringel Morris, Jaime Teevan, and Katrina Panovich. What do people ask their social networks, and why?: A survey study of status message Q&A behavior. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, pages 1739–1748, Atlanta, Georgia, USA, 2010. ACM. ISBN 978-1-60558-929-9. URL http://doi.acm.org/10.1145/1753326.1753587.

[104] Yuanping Nie, Jiuming Huang, Zongsheng Xie, Hai Li, Pengfei Zhang, and Yan Jia. NudtMDP at TREC 2015 LiveQA track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/NUDTMDP-QA.pdf.

[105] Douglas W. Oard and William Webber. Information retrieval for e-discovery. Foundations and Trends® in Information Retrieval, 7(2–3):99–237, 2013. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000025.

[106] Anne Oeldorf-Hirsch, Brent Hecht, Meredith Ringel Morris, Jaime Teevan, and Darren Gergle. To search or to ask: The routing of information needs between traditional search engines and social networks. In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW '14, pages 16–27, Baltimore, Maryland, USA, 2014. ACM. ISBN 978-1-4503-2540-0. URL http://doi.acm.org/10.1145/2531602.2531706.

[107] Aditya Pal and Scott Counts. Identifying topical authorities in microblogs. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pages 45–54, Hong Kong, China, 2011. ACM. ISBN 978-1-4503-0493-1. URL http://doi.acm.org/10.1145/1935826.1935843.

[108] Aditya Pal and Joseph A. Konstan. Expert identification in community question answering: Exploring question selection bias. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pages 1505–1508, Toronto, Ontario, Canada, 2010. ACM. ISBN 978-1-4503-0099-5. URL http://doi.acm.org/10.1145/1871437.1871658.

[109] Aditya Pal, Rosta Farzan, Joseph A. Konstan, and Robert E. Kraut. Early detection of potential experts in question answering communities. In Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization, UMAP'11, pages 231–242, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22361-7. URL http://dl.acm.org/citation.cfm?id=2021855.2021876.

[110] Aditya Pal, Fei Wang, Michelle X. Zhou, Jeffrey Nichols, and Barton A. Smith. Question routing to user communities. In Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management, CIKM '13, pages 2357–2362, San Francisco, California, USA, 2013. ACM. ISBN 978-1-4503-2263-8. URL http://doi.acm.org/10.1145/2505515.2505669.

[111] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 311–318, Philadelphia, Pennsylvania, 2002. ACL. URL http://dx.doi.org/10.3115/1073083.1073135.

[112] Sharoda A. Paul, Lichan Hong, and Ed H. Chi. Is Twitter a good place for asking questions? A characterization study. In Proceedings of the 5th International AAAI Conference on Weblogs and Social Media, ICWSM '11, 2011. URL http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2813/3225.

[113] Virgil Pavlu, Shahzad Rajput, Peter B. Golbus, and Javed A. Aslam. IR system evaluation using nugget-based test collections. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, pages 393–402, Seattle, Washington, USA, 2012. ACM. ISBN 978-1-4503-0747-5. URL http://doi.acm.org/10.1145/2124295.2124343.

[114] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP '14, pages 1532–1543, Doha, Qatar, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[115] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. I wish I didn't say that! Analyzing and predicting deleted messages in Twitter. CoRR, abs/1305.3107, 2013. URL http://arxiv.org/abs/1305.3107.

[116] Martin F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-454-5.

[117] Hemant Purohit, Carlos Castillo, Fernando Diaz, Amit Sheth, and Patrick Meier. Emergency-relief coordination on social media: Automatically matching resource requests and offers. First Monday, 19(1), 2013. ISSN 13960466. URL http://firstmonday.org/ojs/index.php/fm/article/view/4848.

[118] Shahzad Rajput, Virgil Pavlu, Peter B. Golbus, and Javed A. Aslam. A nugget-based test collection construction paradigm. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, pages 1945–1948, Glasgow, Scotland, UK, 2011. ACM. ISBN 978-1-4503-0717-8. URL http://doi.acm.org/10.1145/2063576.2063861.

[119] C. J. Van Rijsbergen. Information Retrieval. Butterworth-Heinemann, 2nd edition, 1979. ISBN 0408709294.

[120] Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, and William R. Hersh. Overview of the TREC 2016 clinical decision support track. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/Overview-CL.pdf.

[121] Adam Roegiest and Gordon V. Cormack. Impact of review-set selection on human assessment for text classification. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, pages 861–864, Pisa, Italy, 2016. ISBN 978-1-4503-4069-4. URL http://doi.acm.org/10.1145/2911451.2914709.

[122] Denis Savenkov. Ranking answers and web passages for non-factoid question answering: Emory University at TREC LiveQA. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/emory-QA.pdf.

[123] Denis Savenkov and Eugene Agichtein. Emory University at TREC LiveQA 2016: Combining crowdsourcing and learning-to-rank approaches for real-time complex question answering. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/EmoryIrLab-QA.pdf.

[124] Milad Shokouhi and Luo Si. Federated search. Foundations and Trends® in Information Retrieval, 5(1):1–102, 2011. ISSN 1554-0669. URL http://dx.doi.org/10.1561/1500000010.

[125] Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. Learning from the past: Answering new questions with past answers. In Proceedings of the 21st International Conference on World Wide Web, WWW '12, pages

759–768, Lyon, France, 2012. ACM. ISBN 978-1-4503-1229-5. URL http://doi.acm.org/10.1145/2187836.2187939.

[126] Luo Si and Jamie Callan. A semisupervised learning method to merge search engine results. ACM Transactions on Information Systems, 21(4):457–491, October 2003. ISSN 1046-8188. URL http://doi.acm.org/10.1145/944012.944017.

[127] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07, pages 623–632, Lisbon, Portugal, 2007. ACM. ISBN 978-1-59593-803-9. URL http://doi.acm.org/10.1145/1321440.1321528.

[128] Ian Soboroff, Charles Nicholas, and Patrick Cahan. Ranking retrieval systems without relevance judgments. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, pages 66–73, New Orleans, Louisiana, USA, 2001. ACM. ISBN 1-58113-331-6. URL http://doi.acm.org/10.1145/383952.383961.

[129] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers on large online QA collections. In Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics, ACL '08, pages 719–727, Columbus, Ohio, 2008. ACL. URL http://www.aclweb.org/anthology/P/P08/P08-1082.

[130] Reem Suwaileh, Maram Hasanain, Marwan Torki, and Tamer Elsayed. QU at TREC-2015: Building real-time systems for tweet filtering and question answering. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/QU-MBQA.pdf.

[131] Reem Suwaileh, Maram Hasanain, and Tamer Elsayed. Light-weight, conservative, yet effective: Scalable real-time tweet summarizatio. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/QU-RT.pdf.

[132] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL '15, 2015. URL http://www.aclweb.org/anthology/P15-1150.

[133] Haihui Tan, Dajun Luo, and Wenjie Li. PolyU at TREC 2016 real-time summarization. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/COMP2016-RT.pdf.

[134] David C. Uthus and David W. Aha. Detecting bot-answerable questions in Ubuntu chat. In Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013, IJCNLP '13, pages 747–752, 2013. URL http://aclweb.org/anthology/I/I13/I13-1089.pdf.

[135] Stalin Varanasi and Günter Neumann. Question/answer matching for Yahoo! Answers using a corpus-based extracted ngram-based mapping. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/dfkiqa-QA.pdf.

[136] Ellen M. Voorhees. The TREC-8 question answering track report. In Proceedings of the Eighth Text REtrieval Conference, pages 77–82, Gaithersburg, Maryland, USA, 1999. URL http://trec.nist.gov/pubs/trec8/papers/qa_report.pdf.

[137] Ellen M. Voorhees. The philosophy of information retrieval evaluation. In Revised Papers from the Second Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems, CLEF '01, pages 355–370. Springer-Verlag, 2002. ISBN 3-540-44042-9.

[138] Ellen M. Voorhees and Donna Harman. Overview of the eighth text retrieval conference (TREC-8). In Proceedings of the Eighth Text REtrieval Conference, pages 1–23, Gaithersburg, Maryland, USA, 1999. URL http://trec.nist.gov/pubs/trec8/papers/overview_8.pdf.

[139] Alexandra Vtyurina, Ankita Dey, Bahareh Sarrafzadeh, and Charles L. A. Clarke. WaterlooClarke: TREC 2015 LiveQA track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/WaterlooClarke-QA.pdf.

[140] Di Wang and Eric Nyberg. CMU OAQA at TREC 2015 LiveQA: Discovering the right answer with clues. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/oaqa-QA.pdf.

[141] Di Wang and Eric Nyberg. CMU OAQA at TREC 2016 LiveQA: An attentional neural encoder-decoder approach for answer ranking. In Proceedings of the Twenty-Fifth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2016. URL http://trec.nist.gov/pubs/trec25/papers/CMU-OAQA-QA.pdf.

[142] Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. A syntactic tree matching approach to finding similar questions in community-based QA services. In Proceedings of the 32nd International ACM SIGIR Conference on Research

and Development in Information Retrieval, SIGIR '09, pages 187–194, Boston, MA, USA, 2009. ACM. ISBN 978-1-60558-483-6. URL http://doi.acm.org/10.1145/1571941.1571975.

[143] William Webber. Approximate recall confidence intervals. ACM Transactions on Information Systems, 31(1):2:1–2:33, January 2013. ISSN 1046-8188. URL http://doi.acm.org/10.1145/2414782.2414784.

[144] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In 4th International Conference on Learning Representations, ICLR '16, 2016.

[145] Guoshun Wu and Man Lan. Leverage web-based answer retrieval and hierarchical answer selection to improve the performance of live question answering. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/ecnucs-QA.pdf.

[146] Lichun Yang, Shenghua Bao, Qingliang Lin, Xian Wu, Dingyi Han, Zhong Su, and Yong Yu. Analyzing and predicting not-answered questions in community-based question answering services. In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI '11, pages 1273–1278, San Francisco, California, 2011. AAAI Press. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3598.

[147] Alexander Yeh. More accurate tests for the statistical significance of result differences. In Proceedings of the 18th Conference on Computational Linguistics - Volume 2, COLING'20, pages 947–953, Saarbrücken, Germany, 2000. ACL.

[148] Reyyan Yeniterzi and Jamie Callan. Analyzing bias in CQA-based expert finding test sets. In Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 967–970, Gold Coast, Queensland, Australia, 2014. ACM. ISBN 978-1-4503-2257-7. URL http://doi.acm.org/10.1145/2600428.2609486.

[149] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pages 587–594, Singapore, Singapore, 2008. ACM. ISBN 978-1-60558-164-4. URL http://doi.acm.org/10.1145/1390334.1390435.

[150] Emine Yilmaz, Evangelos Kanoulas, and Javed A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In Proceedings of the

31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pages 603–610, Singapore, Singapore, 2008. ACM. ISBN 978-1-60558-164-4. URL http://doi.acm.org/10.1145/1390334.1390437.

[151] Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In NAACL-HLT, pages 901–911, 2015.

[152] Hong Yu and Carl Sable. Being Erlang Shen: Identifying answerable questions. In Proceedings of the IJCAI'05 Workshop on Knowledge and Reasoning for Answering Questions, Edinburgh, Scotland, UK, 2005.

[153] G.U. Yule. An Introduction to the Theory of Statistics. Griffin's sci. series. C. Griffin, limited, 1911.

[154] Oren Zamir and Oren Etzioni. Grouper: A dynamic clustering interface to web search results. In Proceedings of the Eighth International Conference on World Wide Web, WWW '99, pages 1361–1374, Toronto, Canada, 1999. Elsevier North-Holland, Inc. URL http://dx.doi.org/10.1016/S1389-1286(99)00054-7.

[155] Weiqian Zhang, Weijie An, Jinchao Ma, Yan Yang, Qinmin Hu, and Liang He. ECNU at TREC 2015: LiveQA track. In Proceedings of the Twenty-Fourth Text REtrieval Conference, Gaithersburg, Maryland, USA, 2015. URL http://trec.nist.gov/pubs/trec24/papers/ECNU-QA.pdf.

[156] Zhe Zhao and Qiaozhu Mei. Questions about questions: An empirical analysis of information needs on Twitter. In Proceedings of the 22nd International Conference on World Wide Web, WWW '13, pages 1545–1556, Rio de Janeiro, Brazil, 2013. ACM. ISBN 978-1-4503-2035-1. URL http://doi.acm.org/10.1145/2488388.2488523.

[157] Liang Zhou, Namhee Kwon, and Eduard Hovy. A semi-automatic evaluation scheme: Automated nuggetization for manual annotation. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, NAACL-Short '07, pages 217–220, Rochester, New York, 2007. ACL.

[158] Tom Chao Zhou, Michael R. Lyu, and Irwin King. A classification-based approach to question routing in community question answering. In Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, pages 783–790, Lyon, France, 2012. ACM. ISBN 978-1-4503-1230-1. URL http://doi.acm.org/10.1145/2187980.2188201.