Answers, not Links

Extracting Tips from Yahoo! Answers to Address How-To Web Queries

Ingmar Weber
Yahoo! Research Barcelona
Avda. Diagonal 177
E-08018 Barcelona, Spain
ingmar@yahoo-inc.com

Antti Ukkonen
Yahoo! Research Barcelona
Avda. Diagonal 177
E-08018 Barcelona, Spain
aukkonen@yahoo-inc.com

Aris Gionis
Yahoo! Research Barcelona
Avda. Diagonal 177
E-08018 Barcelona, Spain
gionis@yahoo-inc.com

ABSTRACT

We investigate the problem of mining "tips" from Yahoo! Answers and displaying those tips in response to related web queries. Here, a "tip" is a short, concrete and self-contained bit of non-obvious advice such as "To zest a lime if you don't have a zester : use a cheese grater."

First, we estimate the volume of web queries with "how-to" intent, which could be potentially addressed by a tip. Second, we analyze how to detect such queries automatically without solely relying on literal "how to *" patterns. Third, we describe how to derive potential tips automatically from Yahoo! Answers, and we develop machine-learning techniques to remove low-quality tips. Finally, we discuss how to match web queries with "how-to" intent to tips. We evaluate both the quality of these direct displays as well as the size of the query volume that can be addressed by serving tips.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

1. INTRODUCTION

For millions of users, search engines constitute the main entry-point to the web, as well as a powerful tool to delve into a wealth of available information. The evolution of web-search engines has been very rapid. Over the years, not only the quality of search results has kept improving, but a large number of novel features has allowed users to find information directly, without the need to click on links away from the search-engine result page. Examples of such features include query recommendations, smart snippets, integration of rich content (images, videos, news, celebrities), direct displays (weather, stock quotes, calculators), and so on.

In this work, we take this approach further by attempting to answer common "how-to" queries *directly*, rather than serving links to sites that contain answers. For example, when a user searches for "how to round decimal in C" we would like to display "To round a decimal in C: add 0.5 and then floor the value – what you call dropping the decimal. Looks like this: int x = (my_float + 0.5);"[1]

We obtain potential answers by mining Yahoo! Answers,[2] the biggest online site for collaborative question-answering. However, rather than looking at all potential answers, including those that are long or those that are opinions rather than concrete solutions, we focus on extracting *tips*. In our context, a tip is a piece of advice that is (*i*) short (160 characters maximum[3]), (*ii*) concrete (it has to be actionable), (*iii*) self-contained (understandable without additional external sources), and (*iv*) non-obvious (otherwise it would be useless). Our focus on tips is mostly motivated by the limited screen real estate, in particular on mobile devices. Furthermore, short and to-the-point tips are often interesting even when they do not perfectly address the underlying problem. Hence, we expect users to be more forgiving than with, say, irrelevant web search results.

To materialize our idea into a functional system we need to address three sub-problems: (*i*) identify when a user query is a how-to query for which it is valuable to display a tip as an answer; (*ii*) extract tips from online resources and populate a large database of high-quality tips; and (*iii*) given a how-to query retrieve relevant tips for that query. Each of the three sub-problems has its own challenges which we we address in this paper. We address them independently creating a modular system, which is easier to build, evaluate, and improve. Figure 1 shows how the different modules connect to each other. The flow of the paper follows the decomposition of our problem into the three sub-problems. In particular, we make the following contributions:

- Using query logs we estimate a *clear how-to intent* in 2.0% of the total search volume of a commercial search engine (Section 3). This is done to demonstrate that directly answering such queries could have a significant

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.

[1] `http://yhoo.it/ly2f1S`
[2] `http://answers.yahoo.com`
[3] 160 characters is the maximum length for an SMS. See `http://latimesblogs.latimes.com/technology/2009/05/invented-text-messaging.html` for an explanation of this limitation.

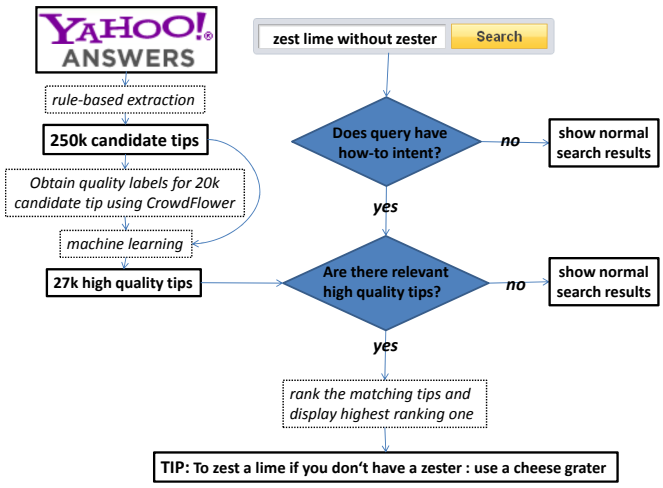

Figure 1: A schematic overview of the system.

impact, as most of the interesting open problems in web IR concern non-navigational queries [28].

- We develop methods to *automatically detect queries with how-to intent*, even when they do not match the literal "how to X" pattern (Section 4). Our best method annotates 1.13% of the total search volume as having a how-to intent. 83.5% of this volume has a clear how-to intent, according to human evaluators.
- We propose a *mechanism to automatically extract tips from Yahoo! Answers* (Section 5). We used a crowdsourcing platform to evaluate 20 000 candidate tips found by our method. 25% of the candidate tips were judged as "very good," 48% as "ok," and 27% as "bad."
- Given the labeled set of tips, we use *machine learning techniques to filter tips of low quality* (Section 5). We obtain a precision of 0.94 (over 10-fold cross-validation) for the best method when separating potentially useful tips from candidates labeled as "bad."
- We *match the set of found how-to queries to the tips labeled by the classifier* (Section 6). The matching algorithm with the highest precision (0.92) returns a tip for 4.5% of the how-to queries. The coverage can be doubled to 10.0%, at the cost of precision dropping to 0.75.

To place our work in the context of question-answering and other related work we survey previous research in Section 7. Finally, we conclude by discussing drawbacks, strengths and future work in Section 8.

2. QUERY DATASET

We used a large sample of a query log of a commercial search engine. The sample contained queries submitted from May 1, 2010 to January 31, 2011. We used only queries by users who had an identifiable cookie related to their account with the search engine, which is also a major email provider. This selection was done to remove a large amount of search spam, typically issued by users who seek to avoid being identified. We also restricted our dataset to queries that led to at least one result click, which is the vast majority of queries. This selection was done since one of our techniques in Section 4 makes explicit use of click information.

The queries were anonymized by replacing potentially sensitive information by special symbols. An inspection of these cases lets us believe that this anonymization does not alter the results in our study as queries which underwent this process are practically never related to how-to intents. Finally, the queries were normalized for case, punctuation, stop-words, and word order.

In Section 4.4 we also made use of click information on a per-host basis and so, for completeness we explain how to obtain hostnames here. To map clicked URLs to hostnames we removed any prefix of the type (`http`|`https`|`ftp`)`://`. If the rest started with `www.`, this token was also removed.[4]

3. PREVALENCE OF HOW-TO QUERIES

Before embarking to solve the problem of finding tips for answering how-to queries, we need to convince ourselves that this is a meaningful problem. So our first question is whether how-to queries make up a large fraction of the search volume.

To answer this question, we sampled a few thousand distinct queries, which were then evaluated by human judges.[5] To obtain better insight whether how-to queries are more prominent among frequent head queries or rare tail queries, we sampled in different ways from the query distribution. The details are presented in the next section.

As far as what actually constitutes a query with a how-to intent, we used the following definition.

> A query is said to have how-to intent if the user is looking for information on solving a particular problem or performing a particular task.

How-to queries are *almost never navigational* and *rarely transactional*. A query such as "online calculator" already implies that the user knows of the existence of relevant online services and is just trying to find any of them. How-to queries are *mostly informational*. A query such as "compute logs in java" would fall into this category.

Judges were asked to evaluate if a query has a how-to intent using a 4-point scale:

2 – clear how-to intent;
1 – possible how-to intent;
0 – no how-to intent;
-1 – foreign language/character set or unintelligible.

The same scale is used in Section 4 where we discuss algorithms to automatically identify how-to queries. Table 1 lists a few queries for these classes.

3.1 Evaluation results

The three judges evaluated three distinct query sets. First, a set obtained by sampling 1 000 queries uniformly from the query volume. This set contains a noticeable fraction of (repeated) head queries. Second, a set obtained by sampling 1 000 queries uniformly from the set of distinct queries. This set has a stronger bias towards the tail of the query distribution than the previous one. Finally, the third set consists of the top 1 000 distinct queries, which are mostly navigational.

[4]These are not necessarily hostnames in a strict technical sense as "bbc.co.uk" and "www.bbc.co.uk" could be served by different hosts.
[5]The authors of this paper.

2: clear how-to	how to install a remington trigger, start a wiki, clean iphone screen, how to hook up tv vcr and dvd
1: possible how-to	copper cleaner homemade, paper match rocket, pizza dough, hair spiking
0: not how-to	my yahoo, mtv, cannon ef, women in war world 2, parkers uniforms houston

Table 1: Examples of queries that are clear how-to ("2"), possible how-to ("1"), and not how-to ("0"). Queries judged as unknown ("-1") largely contained non-Latin characters and are not displayed.

Metric	unif. volume	unif. distinct	top volume
cov. vol.	2.0/1.4/96.3	2.9/3.6/91.6	0.0/0.9/99.0
cov. dist.	2.3/1.7/95.7	3.6/5.2/90.1	0.0/1.2/98.7
agreement	93.9	88.4	96.6

Table 2: Percentage of queries falling into the classes "clear how-to query"/"possible how-to query"/"not a how-to query" when queries are (*i*) sampled uniformly at random over all occurrences, (*ii*) sampled uniformly at random over all distinct queries, or (*iii*) the most frequent 1 000 queries. "Agreement" is measured among the three judges as macro-average over all three pairs.

Each set of 1 000 queries was then randomly divided into one set of 100 and three sets of 300. Each of the three judges evaluated 400 queries: the set of 100 and one of the sets of 300. The common set was used to evaluate agreement among the judges. In computing the agreement, a negligible number of cases where at least one judge selected -1 (foreign language/character set or unintelligible) was skipped and not counted as disagreement. The results are shown in Table 2. We note that the fraction of strong disagreements, 2 vs. 0, was only a tiny fraction of all disagreements.

Overall, 2.0% of query volume and 2.3% of distinct queries were identified as having a clear how-to intent. These fractions are 0% for the most frequent queries though, but higher when sampling from the tail As previous work shows that the frequent queries, which are typically navigational, are "solved" by all major web search engines [28], we believe that a total of 2.0% is actually significant when it comes to giving search engines an opportunity to differentiate themselves from their competitors.

4. DETECTING HOW-TO WEB QUERIES

In the previous section, we showed that how-to queries make up a considerable part of the non-trivial search volume. In this section, we discuss different approaches to detect how-to queries automatically.

4.1 The obvious approach

The most basic approach to extract queries with a how-to intent is to look at queries starting with one of the literal strings "how to," "how do i," or "how can i." Though the precision for such an approach is very high, the recall will be as low as only 1.0% of query instances.

Examples: "how to safely extinguish a campfire," "how do you fix keys on a laptop," "how to do hair like audrey hepburn," "how to train your dragon," "how do you make your basement smell better," "how do i convert video formats."

4.2 Action queries

More experienced users are less likely to enter their information need in natural language with a "how to" prefix. Still, how-to queries could potentially be detected as they refer to an *action* and are likely to contain, or even start with, a verb. Hence, as another approach, we looked at all queries starting with an English verb in its infinitive. Queries were split on non-word characters and the first token was looked up in a verb dictionary.[6] Note that certain words such as "dream" or "sound" are both verbs and nouns.

Examples: "play my music on tool bar raido," "make your own oversized bag," "type of people who put christmas wreaths on cars," "dream symbols," "sound," "buy drawer handles."

4.3 Generalizing literal "how to" queries

As the simple baseline using literal "how to" queries has near-perfect precision, we try to come up with ways to improve its recall, possibly by trading a bit of precision. Concretely, from a query such as "how to fix a flat tire" we want to derive that the query "fix a flat tire" also has a how-to intent. Thus, for all the (literal) "how to" queries we extract, we drop the "how to" prefix, and consider the resulting normalized queries possibly having a how-to intent as well.[7]

However, this normalization alone is overly aggressive. For example, the query "google" is recognized as having a how-to intent, as our query log also contains the query "how to google." To avoid such incorrect generalizations, we look at the fraction of times the normalized query appears with and without the literal "how to" prefix. For example, the normalized query "lose weight" appears as part of a literal "how to" query in 80% of all occurrences. On the other hand, for the normalized query "google," the same fraction is roughly 0.0001%. We experimented with a minimum threshold of 1% (General-.01) and 10% (General-.10).

Examples: "how to dry corn," "fit baseball glove," "stop computer script," "craft ideas for boys," "how to make chocolate bars," "tune your guitar online."

4.4 Using typical how-to answer sites

One drawback of the previous approach is that it only recognizes queries that have been issued at least once as a literal "how to" query. To overcome this limitation we also tried to identify typical how-to *web sites*, such as `http://www.ehow.com/` or `http://www.wikihow.com/`. In analogy to the previous approach, for each clicked host name we looked at the fraction of literal "how to" queries leading to a click on this particular host name. Again we look at the *fraction* and not the *volume* of queries. For example, `http://www.youtube.com/` attracts a large volume of the literal "how to"

[6]Verbs used: `http://www.englishclub.com/vocabulary/regular-verbs-list.htm` and `http://www.englishclub.com/vocabulary/irregular-verbs-list.htm`.

[7]Whenever we refer to the literal "how to" string, we actual mean any of "how to," "how do i," and "how can i."

how-to hosts with largest search volume
wikihow.com
getridofthings.com
howtogetridofstuff.com
dragoart.com
howcast.com
hosts attracing almost exclusively how-to questions
deletepermanently.howto4pc.org
how-to-flirt.org
howtolosefatfast.org
howtogettallersecrets.com
askmethat.com

Table 3: Top how-to host names.

Method	Cover.	Top 500	Rand 500
Baseline	1.01%	99.0/0.0/1.0 (100)	95.8/3.5/0.7 (92)
Verb-first	3.22%	2.3/6.2/91.5 (82)	7.3/7.0/85.7 (83)
General-.10	1.13%	83.5/9.8/6.8 (90)	87.3/6.3/6.3 (89)
General-.01	1.48%	33.8/14.2/52.0 (82)	81.5/9.2/9.2 (86)
Host-.50-20	0.08%	3.3/2.0/94.2 (97)	61.3/14.8/22.8 (61)
Host-.25-10	0.88%	0.0/0.7/99.3 (93)	29.4/16.1/54.2 (64)

Table 4: Experimental results for the detection of how-to queries. The three numbers in the last two columns refer to the percentage of (distinct) queries voted "2", "1" and "0" respectively. Top 500 refers to the most frequent queries labeled as "how-to" by the particular method. The macro-averaged inter-judge agreement is given in parentheses.

queries but it is not a how-to site, as only 1% of queries leading to a click on it are literal "how to" queries.

To identify how-to hosts, we experimented with two different parameters: (*i*) a minimum threshold on the fraction of traffic-generating queries that are literal "how-to" queries, and (*ii*) a minimum threshold on the total number of clicks leading to a particular site.

Once how-to hosts were identified, we looked at their all incoming queries. From these, navigational queries such as "wikihow.com" were removed using a simple heuristic. This heuristic judged a query as navigational (for a given click) if all the tokens in the (normalized) query were also present in the host name. Here we allowed an edit distance of 1 to accommodate minor spelling mistakes. All remaining, non-navigational queries were labeled as "how-to." Host-.50-20 refers to a minimum how-to fraction of .50, where the actual fraction had to *surpass* this threshold, and a minimum traffic count of 20 for the host. Host-.25-10 is defined similarly.

Examples: "xbox 360 delete cache," "fixing a wet cell phone," "fried rice," "aspergers," "how to french braid," "what date is it right to make out."

4.5 Experimental evaluation

We evaluated the task of detecting how-to queries as a classification task. We used the sample of the query log described in Section 2. Our design principle is to put more emphasis on precision rather than on recall. Evaluating the exact recall would require annotating *all* distinct queries with ground truth labels, which is prohibitively expensive. Instead we report the fraction of total search volume that is labeled by a particular approach in the "Coverage" column in Table 4. To put the Coverage figures in perspective, we remind the reader that in Section 3 it was showed that about 2.0% of total web search traffic has a clear how-to intent. We evaluated the precision both for the 500 highest search volume queries labeled by each method, as well as for a set of 500 queries sampled uniformly among the distinct queries detected by this method. Each set of 500 was split into one set of 50 and three sets of 150. Each judge evaluated the set of 50 and one set of 150. The set of 50 was used to estimate inter-judge agreement. The same set of labels $(2, 1, 0, -1)$ as described in Section 3 was used.

As can be seen from Table 4, the simple baseline using the literal "how to" string performs at a reasonable level. Using the General-.10 approach, the coverage can be improved by 13% while still maintaining an acceptable level of precision. It is also worth noting that the precision of the verb-first approach could potentially be improved by removing navigational queries such as "chase online" or "face." Of course, there is also a large number of other applicable techniques for identifying "related queries," e.g., in the context of query suggestions [4, 15]. However, due to the fact that (*i*) the how-to detection is completely modular and (*ii*) the performance of General-.10 is sufficient, we decided to postpone evaluation of more involved schemes to a later stage.

We also considered alternative approaches for detecting how-to intent, such as using online sources, and in particular web-search results. We looked at result snippets and the page content to see if the query matches a "how to" string. Preliminary work showed that this approach works well for popular how-to queries, but as such queries are also detected by other techniques we decided to postpone this approach.

5. MINING TIPS FROM YAHOO! ANSWERS

The next step is to identify a set of potential answers, or tips. A tip is a piece of advice that is (*i*) short (160 characters maximum), (*ii*) concrete (it has to be actionable), (*iii*) self-contained, and (*iv*) non-obvious. The editorial guidelines used for evaluating tips are described in Section 5.3.

We chose to consider tips that have a specific structure "X : Y." Both parts X and Y are short pieces of text. X defines the *goal* of the tip, i.e., a specific situation for which ones requires an actionable solution. Y defines the *suggestion* of the tip, i.e., a proposed solution. The reason that we require tips to have the above structure, is because we find it expressive, easy for the user to parse and understand, and easier for our algorithms to extract and match to queries. As we mentioned before, tips are extracted from Yahoo! answers. Given a ⟨question, answer⟩ pair from Yahoo! answers, the 'question' is used to construct the goal X and the 'answer' to construct the suggestion Y.

Section 5.1 describes the data set from which we extracted our tips. Section 5.2 discusses the steps that were taken to construct candidate tips. In Section 5.3 we give details of the Crowdsource evaluation used to obtain ground-truth quality labels for a subset of candidate tips, which were then used as training data for machine-learning techniques to filter out low-quality tips. Details are given in Section 5.4. The performance results for this classification task are then discussed in Section 5.5.

5.1 Yahoo! answers dataset

We used data from Yahoo! Answers to extract candidate tips. Concretely, we used all ⟨question, best answer⟩ pairs, where the question was posted in the period between May 2006 and August 2010. Best answers are decided either (*i*) by the asker, or (*ii*) by popular vote among Yahoo! Answers users if the asker allows this, or (*iii*) automatically by the system after a certain amount of time. We only used *best* answers as we were interested in extracting high-quality tips, while potentially sacrificing coverage. We further filtered the data by requiring that the user who submitted the question used the English-language front-end.

For questions, only the title was used; the additional and optional information provided in the question description was ignored. Similarly, for answers, we did not use the optional reference field, which, when not empty, contains a justification for the given answer. We did, however, keep track of its binary presence/absence and this was later used as a feature in the machine-learning algorithms. Yahoo! Answers also comes with a topical hierarchy and we recorded the first two levels of this hierarchy for every ⟨question, answer⟩ pair.

We did *not* use any information about user points, which are awarded for contributing answers, nor did we attempt to detect key contributors. This was a deliberate choice in order to make our approach applicable to other sites where the contributor of a potential tip might not even be known.

5.2 Filtering and constructing tip candidates

From the initial set of ⟨question, answer⟩ pairs we constructed a set of candidate tips as follows.

First, we *check that both the question and answer are in English.* Even though we only considered ⟨question, answer⟩ pairs where the question was submitted through an English front-end to Yahoo! Answers, there was still a noticeable fraction of questions and answers in languages such as Spanish, or languages spoken in India. To filter out non-English content, we applied a stop-word-based language-detection technique [13]. Concretely, we used the following list of English stop words: *a, an, and, are, as, has, have, i, in, is, it, me, my, not, of, or, that, the, they, to, was, we, were, will, with, you, your.* Any ⟨question, answer⟩ pair that contained less than three instances of these tokens was removed from further consideration.

Second, we *only accept literal "how to" questions.* We only considered ⟨question, answer⟩ pairs where the question started with "how to," "how do i," or "how can i." This was done for three reasons: First, such questions are typically related to an actual problem or task, as opposed to opinion questions such as "Why have our country's politicians moved so far to the extreme right?"[8] or general knowledge questions such as "How do wildfires affect the ozone layer?"[9] Second, the tips generated from such questions can later more easily be matched against similarly structured web queries. Third, limiting the scope to questions of the given structure made it possible to convert passive "how to X?" questions into active "To X : . . . " pieces of advice. We also removed *multi-part questions.* Some questions, such as "Does the truth set you free, or does it simply hurt your feelings? And how do you (usually) handle it?"[10] are made of several sentences, which make the construction of tips very hard. We used simple punctuation rules to keep only single-sentence questions.

Third, we *required that the answer starts with a verb.* We observed that concrete pieces of advice typically start with a verb in the imperative such as "Try this," "Go there," or "Do that." Remember again that our goal is to maximize precision, with a possible sacrifice in the coverage. We compiled a dictionary of English verbs and we used only the infinitive of the verbs. Conveniently, in English the imperative form of a verb agrees with its infinitive without the "to." Answers starting with "do you" or "have you" were ignored, even though both of these verbs were in our dictionary.

We also experimented with a part-of-speech (POS) tagger,[11] to identify answers where the first token of an answer was used as a verb. The performance of this POS-based approach was very similar to the simple method described above. In fact, due to malformed sentences, the POS-based approach incurred a slightly lower recall. Hence, we decided to stick with the simpler approach. We hypothesize that for textual corpora that contain more "proper" English and less online lingo and typos, POS-based methods and other more "semantic" approaches are likely to outperform the dictionary-based approach. Note that the use of an English verb dictionary provides additional filtering for non-English questions or answers.

After these filtering steps, we are left with a set of ⟨question, answer⟩ pairs to which we apply some syntactic transformations to the question, and construct a tip "$X : Y$" by concatenating the transformed question with the answer. The *tip goal* X is the transformed question, and the *tip suggestion* Y is the answer of the given ⟨question, answer⟩ pair.

To transform the question, we first *convert the questions to active advice* by replacing the initial "how (to | do i | can i)" in the question by a simple "to." Moreover, we *transform the question from ego-centered to user-centered.* Questions asked on Yahoo! Answers are often personal in nature and users bring themselves into the question by making use of first person pronouns such as *I, my, me* and so on. To turn an ego-centered question ("How can I improve my X?") into a user-centered tip ("To improve your X: . . . ") we replaced all occurrences of *I, my, me, am, 'm, myself, our, ours and we* with *you, your, you, are, 're, yourself, your, yours and you.* This replacement changed the tone of the tip from a dialogue to a more objective piece of advice.

Finally, any candidate tip longer than 160 characters was removed. There is a lot of motivation for doing this. First, it makes it easier to display our tips on mobile devices. Second, even on larger screens the "real estate" of a search result page, is very limited. And third, we believe that users will prefer to-the-point bits of advice, when they exist, over long and intricate solutions.

Note that we did not consider information about the age or recency of a tip. In some domains this might be a relevant signal though as, say, the answer to "how to change the default font in Microsoft Word" depends on the version of the corresponding software which, in turn, depends on the time when the query was issued and when a corresponding tip was created.

Example: The question-answer pair "Q: How can I get the mildew smell out of my towels? – A: Try soaking it

[8] http://yhoo.it/kR31Z4

[9] http://yhoo.it/mf5NX0

[10] http://yhoo.it/ifn0jv

[11] http://search.cpan.org/~acoburn/Lingua-EN-Tagger/Tagger.pm

	Very good	Ok	Bad
Very good	12.4%	22.9%	4.8%
Ok	–	25.7%	17.5%
Bad	–	–	16.6%

Table 5: Judge disagreement as computed by macro-averaging over all (judgment, judgment) pairs on a per-tip basis.

in a salt water solution, then washing with soap and cold water, that tends to get rid of smells."[12] was converted to the potential tip "To get the mildew smell out of your towels : try soaking it in a salt water solution, then washing with soap and cold water, that tends to get rid of smells."

5.3 Judging the quality of the tip candidates

The process described above generated a total of 249 675 candidate tips. However, these candidate tips were of a very mixed quality. In order to filter out tips of bad quality, we set to build a classifier for tip quality. To obtain training data for the classifier, we used the CrowdFlower[13] interface to Mechanical Turk.[14]

The workers were asked to label the quality of 20 000 candidate tips as "very good," "ok," or "bad." Those 20 000 tips were sampled uniformly at random from the set of 249 675 tips we constructed. The workers were given the following instructions.

"Very good" : The tip should be concrete and non-obvious. It may offer a likely solution for an issue.

"Ok" : The tip could be an opinion, it could be borderline obvious or its correctness is doubtful.

"Bad" : The tip can be unintelligible, joke advice, insulting, obviously wrong or crucial details are missing.

In addition to these short instructions, which were shown above each tip to evaluate, there was also a similar set of more explicit instructions which included four example for each of the three classes.

One of the advantages of CrowdFlower are so called "gold units," which in our case are tips for which the correct quality judgment is known in advance. These "gold units" are shown the workers at random intervals. If the workers' answers do not match the known correct response their trust score deteriorates. If the trust score becomes too low, the workers are excluded from further judgments and their previous judgments are ignored. To remove potential spammers or low quality workers we used a total of 78 "gold units" (22 very good, 21 ok, 35 bad).

We obtained at least 3 trusted judgments for each of the sampled 20 000 candidate tips. Applying a majority voting scheme, 25% of candidate tips were judged "very good," 48% "ok," and 27% "bad." Table 5 gives details about the inter-judge agreement. Note that there was very little disagreement of the type "very good" vs. "bad" but a substantial amount of disagreement between "very good" vs. "ok."

[12] http://yhoo.it/ClKuN

[13] http://crowdflower.com/

[14] https://www.mturk.com/mturk/welcome

5.4 Detecting high-quality tips using machine learning

Our next step is to use the ground-truth labels we obtained as training data in order to automatically detect high-quality tips. As we were mostly interested in having a high-precision classifier, we phrased the problem as a binary classification problem: can we separate the "very good" tips from the "ok" and "bad" tips? We chose this approach as we were less concerned about dropping a large fraction of "ok" tips than showing "bad" tips.

In the experiments we used standard machine learning algorithms: a support-vector machine with the RBF kernel,[15] a support-vector machine with a linear kernel, a decision tree,[16] and a simple k-nearest-neighbor classifier with Euclidean distance [3]. We made an effort to tune the parameters of each method to deliver best possible performance. Both the SVM and decision tree were configured to penalize more for misclassified instances of the negative (i.e., not "very good") class during training to increase precision.

We consider three types of features. The first feature set ("terms") is the tip-term matrix. This is a sparse matrix containing the number of times a term appears in a tip. Only terms occurring in at least 10 and at most 100K tips were included. Due to the high-dimensionality this set of features could not be used for all machine learning algorithms.

The second feature set ("topics") is derived from the first one. Here, we use the first singular vectors [25] of this matrix as the set of features. These can be interpreted as "topics" of a latent semantic space. More formally, if T is the $n \times m$ tip-term matrix, we have $T \approx U_h \Lambda_h V_h$, where h denotes the number of singular values to use, U_h is an $n \times h$ matrix, Λ_h is $h \times h$, and V_h is $h \times m$. The topic-features are given by the columns of the matrix U_h. In the experiments we set $h = 100$.

The last set ("textual") consists of various properties of the question and answer, such as reading level and sentiment. A few of these are specific to the question answering service, but the majority are easy to compute from any text snippet. To assess the quality of writing, we use the Flesch-Kincaid reading level[17] as computed by the "style" unix command.[18] The sentiment of the tip, both positive and negative, is computed using the SentiStrength[19] tool [24]. See Table 10 for further details.

5.5 Experimental evaluation

We evaluated all 9 combinations of the three classification algorithms and three sets of features. Performance is measured with precision and recall in two variants, hard and soft. The hard variant measures how well the classifier separates the class "very good" from the others. With the soft variant the class "ok" is also considered as relevant. All reported performance numbers are averages over stratified 10-fold cross-validation.

As can be seen in Table 7, differences among the classifiers and feature sets are rather small. All methods achieve a precision of at least 0.62 with at least one of the feature sets.

[15] LIBSVM: www.csie.ntu.edu.tw/~cjlin/libsvm/

[16] The `classregtree` function of Matlab's Statistics Toolbox.

[17] http://en.wikipedia.org/wiki/Flesch-Kincaid_readability_test

[18] http://www.gnu.org/software/diction/diction.html

[19] http://sentistrength.wlv.ac.uk/

Very good
To zest a lime if you don't have a zester : use a cheese grater
To round a decimal in c : add 0.5 and then floor the value - what you call dropping the decimal. looks like this: int x = (my_float + 0.5);
To access temporary internet files in windows vista : try clicking on start, run and then type in: %temp% and then press enter. cheers
Ok
To eliminate chapped lips : water water water, drink water alot, 8 and more glasses a day i use vaseline gel. it's the best for me!
To forget about a horror movie : sing a song over and over again to stop your mind from thinking about it
To prevent getting gastro : wash hands frequently and don't put them in nose/mouth/eyes.....
Bad
To make money at the age of 15 : hook up with your friends moms.
To make your computer super fast : tie it to the hood of your car and take it on the interstate
To you type a backwards e : use a 3. 3 look's like it

Table 6: Examples of candidate tips for the three quality classes. Poor quality tips are later filtered out by machine learning techniques (see Section 5.4).

	Method	terms	topics	textual
Hard	SVM		0.62/0.36	0.56/0.24
	linear SVM	0.61/0.39	0.67/0.30	0.59/0.16
	tree		0.63/0.13	0.60/0.08
	k-NN	0.65/0.14	0.64/0.20	0.57/0.18
Soft	SVM		0.95/0.18	0.93/0.13
	linear SVM	0.94/0.20	0.96/0.14	0.93/0.08
	tree		0.96/0.07	0.93/0.04
	k-NN	0.95/0.07	0.95/0.10	0.94/0.10

Table 7: Precision/recall for binary "very good" vs. "not very good" classification of potential tips. In the "soft" setting the evaluation was for "very good or ok" vs. "bad" for the *same* binary "very good" vs. "not very good" classifiers.

This is a good result, given that the fraction of the class "very good" is roughly 0.24, and that recognizing good tips can be a difficult problem even for humans. In the case of soft evaluation all methods reach at least 0.94 precision, albeit recall is now even lower. However, it must be emphasized that when tips are shown together with web search results, it is better to not show a tip than showing a tip of low quality. For this study we chose linear SVM with terms features as the combination to be used later when filtering good tips (a total of 26.7k) from our set of candidate tips.

Table 8 shows the hard evaluation across different categories in Yahoo! answers. (Note that these are computed from the training set!) First, the number of tips found in a category varies a lot. Categories rich in possible tips are e.g., "Computers & Internet" and "Family & Relationships," while tips about "Business & Finance" or "Society & Culture" are more rare. Also the fraction of "very good" tips varies considerably across the categories. Finally, the classifier performs a lot better in some categories than others. For example, despite the total fraction of tip candidates, as well as the fraction of "very good" tips being rather low in the "Entertainment & Music" category, the classifier achieves 0.63 precision. In summary, the difficulty of the classification task clearly depends on the topic of the tip in question.

6. MATCHING TIPS TO WEB QUERIES

As a final step, we discuss how to put together the components we described in the previous sections in order to complete the design of our system. In particular, in this step we show (*i*) how to decide which how-to queries to consider, (*ii*) how to find tips for those queries, and (*iii*) how to rank those tips. At heart, this is an information-retrieval task, however, care should be taken in order to account for the special structure of tips, their short length, and the fact that displaying a tip is optional and subject to quality.

Category	P,R	VG	size
Beauty & Style	0.52,0.06	0.16	0.08
Business & Finance	0.48,0.17	0.20	0.03
Cars & Transportation	0.81,0.11	0.23	0.03
Computers & Internet	0.69,0.44	0.45	0.15
Consumer Electronics	0.65,0.29	0.38	0.06
Entertainment & Music	0.63,0.32	0.15	0.05
Family & Relationships	0.64,0.05	0.06	0.14
Games & Recreation	0.65,0.34	0.24	0.04
Health	0.52,0.05	0.15	0.09
Home & Garden	0.39,0.05	0.27	0.04
Society & Culture	0.62,0.19	0.09	0.03
Sports	0.81,0.15	0.19	0.03
Yahoo! Products	0.73,0.53	0.45	0.07

Table 8: Category specific P(recision)/R(ecall) (hard) in training data for the 13 categories with the most tips, using a linear SVM with 100 topic features. VG refers to the fraction of labeled "very good" tips in the training data. Size refers to the total fraction of all identified high quality tips falling into the corresponding class.

6.1 Deciding which web queries to consider

Which queries should we consider as candidates for showing a tip? We decided to be fairly conservative and to even consider showing a tip only if the query was judged as a how-to query according to the "General-.10" approach (Section 4.3). Our motivation was that for such queries the tip could answer directly the underlying information need.

Note, however, that in many cases it might be interesting to show "related tips," even when no how-to intent is present. For example, for the query "sour cream" one could display the tip "To make fresh homemade sour cream : add a few drops of lemon juice to double cream."[20] This type of display would have more of a "Did you know . . . ?" flavor. It would

[20] `http://yhoo.it/meI2YV`

be interesting to evaluate through user studies how open users are for such kinds of serendipitous bits of advice.

6.2 Deciding which tips to consider for a query

Assume that we have decided to consider a query as candidate for showing a tip. The next question is to find a set of tips that could potentially be displayed in response to the query. Note that this binary decision of "Should I show anything at all?" does not arise in traditional web search where the search engine is *obliged* to return matching results, if any, even if they are of low quality.

For choosing a minimum "relatedness" threshold, there are many choices. In a strict scheme, for a query "how to do X" only tips of the form "To do X : ..." would be considered. In this setting the query equals the goal of the tip.

Though this conservative approach has good precision (.72-.92 depending on the evaluation setting, see Table 9), it suffers on recall as it is overly conservative. To allow fuzzy matches and improve recall, we experimented with two parameters. First, we experimented with both AND and OR modes. In the AND mode *all* terms in the (normalized) query have to appear in the *goal* part of the tip. In the OR mode, partial matches are also counted. The first column in Table 9 refers to the matching mode. Second, we experimented with a parameter of *matching span*, which is defined as the fraction of distinct tokens in the tip goal that appear in the query. So, a minimum span of 0.5 requires at least half of the tokens in the tip goal to be in the query. The second column in Table 9 refers to this parameter.

We also experimented with matching using the *whole* text of the tip, not just the goal. However, this led to displaying tips that were only weakly related to the queries. We did not experiment further with this approach as we were interested in displaying answers to information needs, rather than "Did you know ...?" bits of advice.

6.3 Ranking tips for a given query

Once a set of candidate tips has been identified for a given query, a myriad of different ranking functions could be tried. Note that we are only interested in displaying a *single* tip and so precision at one (P@1) is a natural measure for the quality of a ranking function.

We used a simple ranking function. We rank according to the number of tokens in the query that are found in the tip goal, normalized by the goal's token length. Repeated tokens in the query are counted repeatedly. Note that stop words were previously removed. Ties were broken using `tf-idf` with simple logarithmic `idf`. Remaining ties were broken by giving preference to longer tips.

We deliberately decided against using more advanced features, such as the confidence of the classifier which labeled the tip as "very good" or signals such as the Yahoo! Answers point count of the person giving the underlying answer. This was done to keep our approach as modular as possible and work with arbitrary textual tips following the "To do X : try Y" structure. We also decided against using `tf-idf` as the primary ranking criteria as preliminary experiments showed that this often led to the preference of a single, rare term being present in both the query and the tip while giving too little importance to tokens such as "ipod" which were very common in our data set. Additionally, there were typically only very few tips to rank due to the minimum span threshold of .5 (see Section 6.2 and Table 9), in particular with an AND mode of query execution.

mode	span	vol.	dist.	P@1	med.	agr.
AND	.50	10.0%	3.2%	.497/.747	2	56%
AND	.66	7.3%	2.2%	.573/.840	2	49%
AND	1.0	4.5%	1.0%	.717/.922	1	63%
OR	.50	78.6%	85.6%	.057/.137	2	87%
OR	.66	33.5%	35.7%	.132/.247	2	83%
OR	1.0	12.2%	10.2%	.187/.394	2	56%

Table 9: The first two columns refer to different settings for the (binary) decision of whether to include a tip as a display candidate for a given query. Volume (vol.) and distinct (dist.) refer to the fraction of General-.10 "how-to" queries for which a tip was found. P@1 is the quality, averaged over 500 distinct (query, highest ranked tip) pairs. For the first number only 2's are counted as relevant, whereas for the second number 1's are also included. Median (med.) is the median number of display candidate tips to rank, when this set is non-empty. The ranking function was the same in all cases and is described in Section 6.3.

6.4 Experimental evaluation

We experimented with a total of six different parameter configurations for determining whether a tip should be shown for a query or not. For each of the six settings, 500 distinct (query, tip) pairs, sampled uniformly at random, were evaluated by three different judges with 50 pairs being evaluated by all three to estimate inter-judge agreement.

Judges were asked to evaluate the quality of the (query, tip) match on a 3-point scale by judging if the tip is ...

2 – ... both closely related to the query and very useful in addressing the user's information need;
1 – ... at least somewhat related to the query and potentially useful in addressing the user's information need;
0 – ... either not related to the query or is not useful in addressing the user's information need.

Results are shown in Table 9. Note that this evaluation does not only include the quality of the (query, tip) matching, but also the quality of the tip itself. Similarly, if the query lacks a how-to intent (e.g., "how to train your dragon") this will also hurt the performance.

7. RELATED WORK

This paper describes a system that (*i*) process a question-answering corpus to create a large database of *tips*, (*ii*) identifies how-to queries from a web query log, and (*iii*) serves those queries with tips. To our knowledge, this is a novel task. In this section we discuss work that is related to various components of our system.

Work on Yahoo! answers. We create a database of tips by processing automatically questions and answers from Yahoo! answers. Then we employ a machine-learning approach to assess the quality of those tips and filter our tips of bad quality. In the spirit of this task, Agichtein et al. [2] build a machine-learning approach to assess quality of questions

and answers in Yahoo! answers. Similarly, Bouguessa et al. [6] apply link-analysis techniques to identify authoritative users in Yahoo! Answers. Both of these works employ user-centered and community-based features. In our work, we made the conscious decision not to consider such features to have our methods more generally applicable. Harper et al. [11] investigate the problem of classiying questions into informational vs. conversational and they observe that informational questions have a higher archival value and are more useful for future reference. They found that personal pronouns are useful features in this distinction and we included these in our feature set (see Table 10). In another line of work, Adamic et al. [1], Surdeanu et al. [22], and Kim and Oh [16] address the problem of finding automatically the best answer to a question in Yahoo! answers. The techniques and features used in all the above-mentioned papers could be potentially applied in our system in order to select more questions and answers of good quality, and thus, to enlarge our database of tips.

Question answering and databases of factoids. Contrasted with traditional information retrieval, where the goal is to retrieve full documents, the problem of *question answering* [18, 26] asks to locate possibly very fine-grained, targeted answers to questions given by the user. At a high level our work also falls in this framework, but there are important differences to existing approaches to question answering. In particular, we are not concerned with constructing an intermediary database of facts for assembling answers to questions, as done e.g., by Hildebrand et al. [12] or Denicia-Carral et al. [10]. Nor is our work about passage retrieval [9, 23], as we are not direcly using passages of text as the tips. A sophisticated passage retrieval system might also be useful to find tips for how-to questions, but so far research on passage retrieval has concentrated on factoid questions, not procedural knowledge, and are thus not directly applicable in our setting. Moreover, our system is inherently designed to work with a keyword search engine, as we can assign a how-to intent also to queries that are not explicitly formulated as a question by the user. Finally, many Q&A systems rely on proper punctuation but our approach also works with imperfect user-generated content.

Also related is the recent paper by Bian et al. [5], where the authors define the problem of *question-answering retrieval*, as the problem of retrieving high-quality question-answer pairs, from a question-answering portal, in response to a web query. This problem is again different than the one we address in this paper in many ways. Most importantly, we focus in how-to queries, and we provide a methodology to identify such queries. Also we combine questions and answers in a concise piece of advice that we call tip.

The idea of automatically extracting information from online sources to build datasets of factoids is another active area of research. A typical task in this line of work is ontolgy construction. Suchanek et al. [21] combine data from Wikipedia and WordNet to automatically build a large scale ontology of real world entities. Another popular application domain is biology, with the objective of assisting curators to mine different types facts from medical publications, in particular entities, and relationships between these entities [8, 27]. An important property that distinguishes our problem is that an ontology or any other database of factoids is unlikely to be sufficient to construct answers how-to queries. Our tips often describe a process that is not easily extracted from relationships between entities.

Query-indent classification. A component of our system is to automatically identify how-to queries. This component is related to a line of work on query-intent classification. Broder [7] defined a classification of web queries into three categories: informational, transactional, and navigational. Rose and Levinson [20] further extended the Broder's taxonomy and introduced goal hierarchy. Following Broder's taxonomy, several authors have focused on the automatic classification and characterization of user intent [14, 17]. In a somewhat related work, Radlinski et al. [19] find popular meanings of queries using web-search logs and user-click behavior. In our work, we do not aim to classify all queries in a hierarchy of intents, instead we are focusing on only identifying how-to queries. Again, we view the line of work on query-intent classification as complementary to our work, as techniques from these papers can be potentially employed to improve the accuracy of our query-identification task.

Existing services. The idea of serving tips for mobile devices with a focus on local/geographical events has been used in systems such as Foursquare[21] and View.[22] Differences between those services and our work are that (*i*) they do not extract tips automatically, and (*ii*) they exclusively focus on tips with a local/geographical context. So, for instance, they would not provide any tips such as "To change the default font in Microsoft Word do ...." The service most closely related to our system is probably *ClueDB*.[23] ClueDB allows users to submit tips (called "clues"), vote, and comment on each other's tips. However, ClueDB does not make use of any automatic extraction of tips but relies solely on a community effort. Even though we believe that a community of tip curators could add significant value, we also claim that automatic extraction of tips is a valuable tool to overcome the cold-start problem. In our work we also address the problem of matching queries to tips and ranking a set of candidate tips.

8. CONCLUSIONS

In this paper, we described a modular system that extracts high-quality tips from Yahoo! Answers and shows them as a direct display for matching web queries. We believe that such a system helps web search engines to differentiate themselves from competitors by providing "Answers, not Links"—an often heard phrase concerning the future of web search.

In the future we hope to improve our system along a number of dimensions. First, each of the individual parts (how-to identification, tip extraction, removal of low-quality tips, selecting candidate tips for web queries, ranking those candidate tips) could be improved by (*i*) incorporating additional features and (*ii*) combining our techniques with existing ones. Second, it seems feasible to extract tips from other data sources such as Twitter or travel sites. For Twitter, tip-related hash tags could be a useful signal. For Travel related and, in fact, arbitrary sites rules such as "Tips have to start with a verb." could be combined with part-of-speech tagging to ensure that the supposed tip is self-contained and does not refer to previous parts of text. Third, we would like to identify other query types which could be addressed by

[21] `http://foursquare.com`

[22] `http://view.io/`

[23] `http://www.cluedb.com/`

(Y! Answers specific) HAS-REFERENCE: Indicates whether the "reference" field in the answer was used. (Contents of this field *were not* used when constructing tips!)
(Y! Answers specific) AWARD-TYPE: Indicates whether the answer was selected "best answer" by the asker, by the community or automatically after a certain time.
WORD-IN-TIP: Occurrence counts for words in the tip that appear in at least 10 and in at most 100,000 tips.
FREQUENT-FRACTION: The fraction of words found in a frequent word dictionary. We used a list of 2 126 distinct words downloaded from `http://www.paulnoll.com/Books/Clear-English/English-3000-common-words.html` to compute the fraction of tokens that could be found in a dictionary of common English words;
READING-LEVEL: Flesch-Kincaid reading level.
POSITIVE-SENTIMENT-[Q|A]: The positive sentiment of the question according to SentiStrength.
NEGATIVE-SENTIMENT-[Q|A]: The negative sentiment of the question according to SentiStrength.
1ST-PERSON-SINGULAR-[Q|A]: The case insensitive count of "i", "my" etc. appearing in the question.
1ST-PERSON-PLURAL-[Q|A]: The case insensitive count of "we", "our" etc. appearing in the question.
2ND-PERSON-[Q|A]: The case insensitive count of "you", "yours" etc. appearing in the question.
3RD-PERSON-SINGULAR-[Q|A]: The case insensitive count of "he", "she", "his" etc. appearing in the question.
3RD-PERSON-PLURAL-[Q|A]: The case insensitive count of "they", "theirs" etc. appearing in the question;
FRACTION-UPPER-CASE-[Q|A]: Fraction of all uppercase characters (A-Z) in the question.
REPEATED-?!-COUNT-[Q|A]: The count of sequences matching /[!?]2,/
POSITIVE-EMOTICONS-[Q|A]: Number of emoticons from the set ":-)",":)", ";-)" and ";)".
NEGATIVE-EMOTICONS-[Q|A]: Number of emoticons from the set ":-(" and ":(".
URL-COUNT-[Q|A]: Count for the regular expression /[a-z]3, .(com|net|org|ca|uk)/ in the question.
TOKEN-COUNT-[Q|A]: Total number of tokens in the question when splitting on all white spaces and sentence punctuation.

Table 10: Textual features. The notation [Q|A] indicates that the feature appears separately for the question and answer parts of the tip.

similar techniques. To identify relevant query classes we envision applying a Jeopardy-like approach to answer the question "What type of web query should this short piece of text be returned to?"

Acknowledgements. This research was partially supported by the Torres Quevedo Program of the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund, and by the Spanish Centre for the Development of Industrial Technology under the CENIT program, project CEN-20101037, "Social Media" `http://www.cenitsocialmedia.es/`.

9. REFERENCES

[1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *WWW*, 2008.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, 2008.

[3] E. Alpaydin. *Introduction to Machine Learning*. 2004.

[4] I. Antonellis, H. G. Molina, and C. C. Chang. Simrank++: query rewriting through link analysis of the click graph. *VLDB*, 1, 2008.

[5] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *WWW*, 2008.

[6] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *KDD*, 2008.

[7] A. Broder. A taxonomy of web search. *SIGIR Forum*, 2002.

[8] A. M. Cohen and W. R. Hersh. A survey of current work in biomedical text mining. *Briefings in Bioinformatics*, 6(1), 2005.

[9] H. Cui, R. Sun, K. Li, M.-Y. Kan, and T.-S. Chua. Question answering passage retrieval using dependency relations. In *SIGIR*, 2005.

[10] C. Denicia-Carral, M. Montes-Gómez, L. Villasenor-Pineda, and R. Hernández. A text mining approach for definition question answering. In *ANLP*, 2006.

[11] F. Harper, D. Moy, and J. Konstan. Facts or friends?: distinguishing informational and conversational questions in social Q&A sites. In *CHI*, 2009.

[12] W. Hildebrandt, B. Katz, and J. Lin. Answering definition questions with multiple knowledge sources. In *HLT/NAACL*, 2004.

[13] N. C. Ingle. A language identification table. *Incorporated Linguist*, 15(4), 1976.

[14] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing Management*, 44, 2008.

[15] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, 2006.

[16] S. Kim and S. Oh. Users' relevance criteria for evaluating answers in a social q&a site. *JASIST*, 60(4), 2009.

[17] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW*, 2005.

[18] V. Moriceau, E. SanJuan, X. Tannier, and P. Bellot. Overview of the 2009 QA Track. *Focused Retrieval and Evaluation*, 2010.

[19] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *WWW*, 2010.

[20] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW*, 2004.

[21] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Journal of Web Semantics*, 6(3), 2008.

[22] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online QA Cllections. In *ACL*, 2008.

[23] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR*, 2003.

[24] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. Sentiment in short strength detection informal text. *JASIST*, 61, 2010.

[25] L. Trefethen and D. Bau. *Numerical linear algebra*. Society for Industrial Mathematics, 1997.

[26] E. Voorhees. Question answering in TREC. In *CIKM*, 2001.

[27] R. Winnenburg, T. Wächter, C. Plake, A. Doms, and M. Schroeder. Facts from text: can text mining help to scale-up high-quality manual curation of gene products with ontologies? *Briefings in Bioinformatics*, 9(6), 2008.

[28] H. Zaragoza, B. B. Cambazoglu, and R. Baeza-Yates. Web search solved?: all result rankings the same? In *CIKM*, 2010.