Learning to Rank Effective Paraphrases from Query Logs for Community Question Answering

Alejandro Figueroa
Yahoo! Research Latin America
Av. Blanco Encalada 2120
4th floor, Santiago, Chile
afiguero@yahoo-inc.com

Günter Neumann
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
neumann@dfki.de

Abstract

We present a novel method for ranking query paraphrases for effective search in community question answering (cQA). The method uses query logs from Yahoo! Search and Yahoo! Answers for automatically extracting a corpus of paraphrases of queries and questions using the query-question click history. Elements of this corpus are automatically ranked according to recall and mean reciprocal rank, and then used for learning two independent learning to rank models (SVMRank), whereby a set of new query paraphrases can be scored according to recall and MRR. We perform several automatic evaluation procedures using cross-validation for analyzing the behavior of various aspects of our learned ranking functions, which show that our method is useful and effective for search in cQA.

Introduction

Community Question Answering (cQA) services such as Yahoo! Answers have become very popular for maintaining and distributing human–made web content in form of textual questions and their answers on a very large scale. A huge crowd of millions of people exchange dynamically and rapidly questions and answers basically about anything, where the particular questions and answers may vary greatly in complexity and relevance from a linguistic as well as content point of view. Today, the largest cQA services maintain over 100 million answered questions which in its own represents a very huge and valuable knowledge base (Zhao et al. 2011). Consequently, the topic of effective search in cQA engines have recently attracted many researchers, e.g., (Zhao, Zhou, and Liu 2007), (Lin 2008), (Zhao, Wang, and Liu 2010), (Suzuki, Nakayama, and Joho 2011), (Zhao et al. 2011), (Zheng et al. 2011).

A promising approach in this direction has been proposed by (Lin 2008). The core idea is to use the user generated questions of a cQA along with search engine query logs to automatically formulate effective questions or paraphrases in order to improve search in cQA. (Zhao et al. 2011) have further elaborated this idea into the direction of generation of new questions from queries. A major advantage of such a query-to-question expansion approach for cQA is that it can help to retrieve more related results from cQA archives and hence, can improve the recall. Similarly (Zhao, Wang, and Liu 2010) proposed an approach for extracting paraphrases from search engine query logs and to use them for generating paraphrase patterns.

In this paper, we present a novel approach of using user generated questions of cQA along with search engine query logs, viz. *to learn to rank generated paraphrases* in order to improve the search for answers in cQA. The motivation behind this is as follows: The automatic generation of paraphrases is a useful means to improve the search for finding best answers in cQA. But the generated paraphrases (although they might "mean" the same) do not necessarily lead to the same answer set, and hence, it might be that they differ in the expected retrieval quality of identifying and ranking best answers high. Thus, it makes sense to rank the generated paraphrases, so as to provide evidence according to recall and the position of the best answer of a paraphrase, i.e., its mean reciprocal rank (MRR).

The core idea of our method is as follows (see Fig. 1). Given a huge collection of query logs from a Yahoo! Search, we extract all pairs consisting of a query and a title, where at least one user click links the query with a title from Yahoo! Answers. We further cluster these pairs into groups, where each group consists of all query-title pairs with same title. We interpret each group (including the title) as a set of paraphrases of the same underlying question. Note that each title is associated with an answer web page (actually, the title is the user entered question of the answer web page), and so also its paraphrases. This way we obtain a huge collection of answer web pages and their associated question paraphrases. In a next step we assign to each paraphrase a recall and MRR value which are automatically computed by querying each paraphrase to all answer web pages. We are now able to automatically sort the paraphrases sets according to recall and MRR independently, cf. table 1 for an example. Furthermore, we can extract features and learn two separate learning to rank (SVMRank) models, one for ranking new paraphrases according to recall and one for ranking them in congruence with MRR.

Through a number of experiments, we will demonstrate that these ranking functions are effective for search in cQA. Furthermore, our experimental results shed light on possible new architectures for cQA engines: On the one hand, question answering systems need to fetch as much query-related content as possible, but on the other hand, context bearing answers must be ranked at the top. Our outcomes actually suggest that a two-step retrieval approach is a better option than a one step straightforward approach. In the first step, the cQA system might benefit from a purpose-built ranking function that boosts recall, and in the second step, it might capitalize on a ranking function focused on MRR that operates on the pre-fetched elements determined in the first step.

Note that we conceive paraphrases in a broad sense, that is we do not explicitly only consider well-formulated questions (e.g., "does lack of iron cause headaches?"), but also implicit requests ("headache iron"), grammatically incorrect queries ("and headach low iron") and other semantic alternatives ("migraine headaches low iron" or "can low hemoglobin cause headaches?"). Note further, that all information stored in a web answer page is retained, which in general not only contains relevant answers, but any comment made by the community for that selected question. We think, that both aspects together are very important to define realistic test cases and to achieve robustness on real cQA data.

Related Work

Query Search Log Analysis in cQA (Zhao et al. 2011) present a method for automatically generating questions from queries for cQA. Their method includes two main steps, i.e., template acquisition and question generation. They collect query-to-question pairs from search engine query logs on which basis the template acquisition extracts question generation templates. For a new query, the most similar templates are selected and a list of questions are created and sorted. Our approach differs from this work in that we rank paraphrases according to their possible search quality using recall and MRR as major sources, where they rank questions according to the quality of being a well-formulated question. Furthermore, they restrict the generated questions to be formulated very similar to user entered questions, where we do not have this restriction. They perform tests on a small-scale corpus of 1000 randomly sampled queries from Baidu query logs, where we perform large-scale experiments on our complete collection using cross-validation.

(Zheng et al. 2011) also propose a method for generating questions from keywords. They present a keywords to question system (called K2Q) that considers both query history and user feedback. They employ an adaptive language model to describe the process of forming questions, and also use automatically induced question templates to create unseen questions. Candidate questions are also sorted by their fluency, i.e., the probability of being seen in a cQA system.

Like (Zhao et al. 2011), their sorting approach is also tailored to prefer candidate questions which bear a higher quality of being a well-formulated human-like question rather than sorting them according to their quality of improving search which is the focus of our research. They also compare the generated questions according to their similarity with user entered questions from a cQA not taking into account the answers. So, it seems that both approaches are suited for "cleaned" cQA archives, but it is unclear whether they can be simply adapted to work with realistic cQA archives containing a lot of trash or unvalidated answers and other posted information, as we do.

Paraphrase Extraction We extract paraphrases from query logs, that is we do not exploit any statistical model to generate them, which gave us a broader sampling of potential candidates. The core motivation is based on the idea that, if some queries result in similar clicks, then the meanings of these queries should be similar, cf. (Wen, Nie, and Zhang 2002).

(Zhao, Wang, and Liu 2010) were the first to propose the extraction of paraphrases from general search engine query logs and to use them for generating paraphrase patterns. They found that when several queries hit the same title, the queries are likely to be paraphrases of each other, as well as, when a query hits several titles, paraphrases can also be found among the titles. They extract three kinds of paraphrases from search logs and combine them into one model: query-title, query-query and title-title paraphrases. Validation of candidate paraphrase pairs is defined as a binary classification problem and performed by a SVM classifier. Human evaluation results on a manually annotated corpus of 5000 query-title pairs from a search log showed that the method is able to yield high qualitative paraphrases. Major differences to our work are that we evaluate the effectiveness of paraphrases, where they focus on validation, and that we explicitly extract query-title pairs, where the title is the question of a corresponding Yahoo! Answers page, and as such our approach is specifically tailored to cQA.

Learning to Rank (Surdeanu, Ciaramita, and Zaragoza 2011) present an approach for ranking "How to"–answers extracted from Yahoo! Answers. They build a positive and negative training collection of question-answer pairs, where a positive pair consists of a user question and the correct answer selected manually by the user. All other answers to that question are considered as negative. In a similar spirit as we do, they define many different feature extractors and use them for training a function used for ranking answers. The major differences to our approach are: they only consider a restricted set of question types (actually only "how to" questions, where we consider all questions); they focus on ranking answers, where we focus on ranking query paraphrases; and they do not explore search engine query logs.

User clicks have shown to provide valuable relevance feedback for a variety of tasks, cf. (Radlinski, Szummer, and Craswell 2010). For instance, (Ji et al. 2009) extracted relevance information from clicked and non-clicked documents within aggregated search sessions. They modeled sequences of clicks as a means of learning to globally rank the relative relevance of all documents with respect to a given query. (Xu et al. 2010) improved the quality of training material for learning to rank approaches via predicting labels using click-through data.

Proposed Method

Fig. 1 displays the major components and control flow of our new method. Central to our approach is the automatic acquisition of the paraphrases corpus. Its major steps are described in sequential order in more detail below, followed by

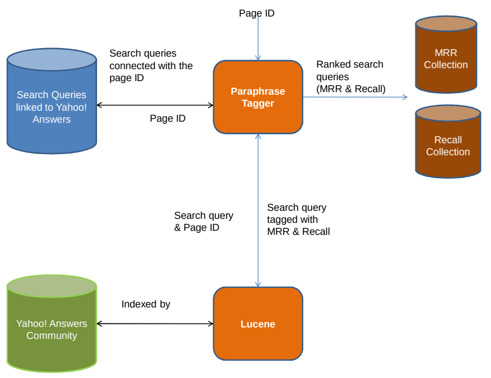
Page ID
Search queries connected with the page ID
Search Queries linked to Yahoo! Answers
Page ID
Paraphrase Tagger
Ranked search queries (MRR & Recall)
MRR Collection
Recall Collection
Search query & Page ID
Search query tagged with MRR & Recall
Yahoo! Answers Community
Indexed by
Lucene

Figure 1: Major components and control flow for both, the training and application phase.

a description of the different features that are used to construct the input for the learner which in our case is SVM-Rank (Joachims 2006).

Query-question collection We first compile a collection of queries submitted to the Yahoo! search engine during the period of January 2011 to March 2012.[1] Since, we are only interested in user queries that can be used to find answers in a cQA (in our case Yahoo! Answers), we only retain those queries which have at least one user click that links the search query with any question in this cQA. Here, we made allowances only for questions posted to this community from June 2006 to December 2011. The question is typically given in the title of the answer page, and it sets the discussion topic of the page. Overall, this step collects 155 million search engine queries corresponding to about 26 million Yahoo! Answers pages.

We interpret a question title and their linked search engine queries as a set of paraphrases of the same underlying question. More specifically, we conceive the title question as the source paraphrase whereas the respective search engine queries as its variations. The source paraphrase is entered by the user when setting the topic of the answer page.

Corpus cleaning Since we noticed that many answers are expressed in languages different from English, we checked every answer and title contained in our collection of 26 million pages. Sometimes the search query is in English, but the related (clicked) Yahoo! Answers web page is, to large extent, in another language. For this purpose, we use a language detector[2], and accordingly, we discarded all content that this tool did not indicate English as a potential language.

Furthermore, given the fact that some questions were duplicated in the community, we merged these instances via title string matching. We also removed all pages connected with more than fifty and less than five paraphrases. Pages linked with high number of paraphrases are not reliable and make the next step too computational demanding, while pages connected with few queries are not likely to provide good and bad reformulations. Note that due to merging, some questions might now have multiple best answers. Here we additionally discarded pages (and their related search queries) that did not have a best answer.

Altogether, this yields a final corpus of about 32 million answers embodied in 6 million pages corresponding to 81 million search engine queries.

Corpus indexing We indexed this pool of 32 million answers with Lucene[3]. When indexing, we removed stop words by means of a list of traditional stop terms extended with some tokens that we identified as community stop words (e.g., “yummy”, “coz”, “lol”, and “y!”). All terms were lowercased. Subsequently, each paraphrase was automatically assessed by sending it to Lucene and checking its recall and the MRR of the highest ranked best answer. The recall was computed by accounting for the number of answers fetched from the related Yahoo! Answers page, or in the event of merged pages, from all the related combined pages. In all these computations, we only considered the top 1,000 hits returned by Lucene.

Recall and MRR collections Now that each paraphrase is automatically annotated with both metrics, we construct a recall and MRR collection as follows:

The **Recall collection** comprises all pages for which we find more than three distinct values for recall across the related paraphrases. Since this rule produced few rankings, we aggregated this set with small rankings (six paraphrases) containing three different ranking values. Eventually, this brought about an increase from 36,803 to 51,148 rankings. The final amount of paraphrases is 814,816.

The **MRR collection** encompasses all pages for which we find more than six distinct values for MRR across the related paraphrases (see example in table 1). This rule selected 54,848 rankings containing 1,195,974 paraphrases.

Note that we now interpret a page as a ranking of paraphrases, more precisely, the search engine queries in conjunction with the title of the respective page. For the reminder of this paper, answers are not longer utilized, and both collections are used separately during feature extraction, training and testing.

Features In our experiments, we took into account the following array of attributes distilled from paraphrases.

Bag of Words (BoW) adds a property to the feature vector representing each term and its frequency within the paraphrase, only considering terms with a global frequency higher than an empirical threshold (see experimental section). Similarly, bi- and tri-gram features are computed.

POS tagging generates features in agreement with their part-of-speech categories.[4] This attribute adds to the feature vector “number-of” attributes: tokens in the paraphrase, tokens tagged as NN, JJ, VB, etc. The “number-of” frequency counts are associated with each paraphrase.

We exploit semantic relations provided by WordNet such as hypernyms (e.g., “hardware $\rightarrow$store”), hyponyms

[1]We only consider English queries, but the whole approach only uses very few language specific resources, so that an adaptation to queries from other language is perfectly possible.

[2]http://code.google.com/p/language-detection/.

[3]http://lucene.apache.org/

[4]Using http://web.media.mit.edu/$\sim$hugo/montylingua/

MRR	pQ	Paraphrase
0.0	F	how manu disc does diablo2 lod come with
0.0	F	how many discs does d2 come with?
0.0018	F	how many disc come with diablo battle chest
0.0018	F	how many disk does diablo battle chest
0.0022	T	How many discs are supposed to be in Diablo II?
0.0124	F	how many discs come with diablo 2 battle chest
0.0130	F	how many disc comes in the diablo 2 box
0.0133	F	how many disk does diablo 2 come with
0.0141	F	how many discs does diablo 2 have?
0.0149	F	diablo 2 number of discs
0.0154	F	how many discs did diablo 2 have
0.5	F	how many disks are in diablo 2 pc?
1.0	F	diablo 2 disks
1.0	F	how many disks does diablo 2 come with

Table 1: An example taken from the MRR collection. pQ stands for posted question. The entries show a ranking consisting of 14 paraphrases and 12 distinct ranking values. The title of the corresponding Yahoo! Answers page is signaled by T, others as F.

("credit→payment"), meronyms ("navy→fleet"), attributes ("high→level"), and regions ("Toronto→Canada"). Similarly to the "number-of" attributes, an element representing the frequency count of the respective type of relation at the paraphrase level is added to the feature vector.

Analogously, we considered collocations provided by the Oxford Dictionary: following (e.g., "meat→rot") and preceding verbs ("consume→meat"), quantifiers ("slab→meat"), adverbs ("steadily→increase"), adjectives ("souvenir→mug"), verbs ("fill→mug"), preposition ("increase→by"), and related noun ("meat→products").

We used eight string distances[5]: jaro, jaccard, jaro-winkler, fellegi-sunter, levenstein, smith-waterman, monge-elkan and scaled-levenstein. For each metric, an additional attribute represents the maximum value between two different tokens in the paraphrase.

Word Lemma is a boolean property indicating whether or not both, a word and its lemma are contained in the paraphrase, e.g. "book" and "books". We used Montylingua for the morphological analysis.

Experiments

The ranking functions are computed using SVMRank, which implements a fast pairwise learning to rank approach (Joachims 2006). In all our experiments, we use five-fold cross validation. In order to maintain consistency across our experiments, we use the same five data random splits. Note our evaluations were carried out on both collections independently: All experiments assessing MRR are conducted on the MRR collection, whereas all experiments evaluating Recall are carried out on the Recall collection.[6]

A key advantage of having all paraphrases annotated in terms of Recall and MRR is that we can compute the **upper bound** for the performance by picking the highest scored element in each ranking. In other words, we can imagine a system or an oracle that always picks one of the best options (see table 1). Hence, the upper bounds for MRR and Recall are 0.41744 and 0.30874, respectively. Obviously, this is the best performance any configuration or system can achieve operating on our two collections. The **lower bound** for the performance is determined by choosing the lowest rated item in each ranking. For our corpus, the lower bounds for MRR is 0.00039, while for Recall it is 0.0073.

Additionally, our collections offer an additional reference for the performance. The title question (source paraphrase) provides a rough approximation of what a *human user* would enter to a QA system (cf. table 1). Notice that the title sets the topic of an answer web page, which is the reference clicked by the users of the search engine. By checking the performance achieved by these titles, we obtain for our corpus: MRR=0.12601 and Recall=0.17979.

We used two **baseline methods** for comparison. The first method, also called BoW(G), uses the BoW approach. We tuned its performance for different thresholds (word frequency counts from 0 to 19). In both cases (MRR and Recall), the optimal threshold was 2, obtaining a performance of MRR=0.1 and Recall=0.1568. Normally, the BoW model yields good performance in many text mining tasks. In our task, it only reached 23.96% of the achievable MRR and 50.79% of the achievable Recall, which is also below the potential human performance.

For the second baseline, we used a centroid vector trained/tested via five-fold cross-validation. We used the same splits of our MRR/Recall collections as our SVMRank models. The vector is composed of terms that appear in at least three paraphrases, where each term is represented by the average MRR/Recall values determined from the retrieved paraphrases. The MRR and Recall values for this baseline are 0.0939 and 0.1543, respectively.

We use a greedy algorithm for performing feature extraction, which starts with an empty bag of features and after each iteration adds the one that performs the best. In order to determine this feature, the algorithm tests each non-selected property in conjunction with all the features in the bag. The procedure halts when there is no non-selected feature that enhances the performance. We refer to the system utilizing the best set of properties discovered by this algorithm as SVMRanker(G).

Table 2 highlights the overall results of the greedy method for both metrics together with the impact of the selected features. Compared to our baseline, our features assisted in enhancing the performance by 8.90% (MRR) and by 4.69% (Recall). This means that these systems finished with 26.08% and 53.17% of the achievable MRR and Recall, respectively.

Our experiments show that most of the Recall and MRR is gained by a combination of two key attributes: 'BoW' and 'Number of NNP' (tokens tagged as singular proper nouns (NNP)), which brings about a growth in performance of 7.92% (MRR) and by 3.97% (Recall). In light of that, we conclude that this configuration leads to a cost-efficient solution. Actually, we examine the data-set and found that the average MRR and Recall of a paraphrase systematically

[5]Using http://secondstring.sourceforge.net/

[6]From now on, all MRR and Recall values refer to the average values obtained when carrying out the cross-validation.

Recall		MRR	
BoW	0.156816	**BoW**	0.099971
Number of NNP	0.163043	**Number of NNP**	0.107887
Bigrams	0.163280	Trigrams	0.108160
Number of JJR	0.163433	**felligi-sunter**	0.108378
Word Lemma	0.163564	**Attributes**$^{+}$	0.108497
Number of SYM	0.163659	**Similars**$^{+}$	0.108607
felligi-sunter	0.163742	Number of RB	0.108649
Ins. Hypernyms^{+}	0.163824	**Number of RP**	0.108682
adverb^{-}	0.163894	**Ins. Hypernyms**$^{+}$	0.108715
Number of NNPS	0.163973	**Sub. Meronyms**$^{+}$	0.108742
Similars^{+}	0.164010	Verb Groups^{+}	0.108770
Sub. Meronyms^{+}	0.164056	Outcomes^{+}	0.108787
Topics^{+}	0.164084	Regions Members^{+}	0.108804
Member Holonyms^{+}	0.164107	Related^{+}	0.108810
Entailments^{+}	0.164129	Ins. Hyponyms^{+}	0.108815
Topics Members^{+}	0.164140	**Number of NNPS**	0.108819
Number of RBS	0.164147	quantifier^{-}	0.108824
Number of RBR	0.164153	**Number of RBS**	0.108854
Number of RP	0.164157	Adjective^{+}	0.108865
Attributes^{+}	0.164161	**Topics**$^{+}$	*0.108871*
Sub. Holonyms^{+}	*0.164163*		

Table 2: Impact of selected features on the performance for the two collections. In the Recall column, e.g., 'BoW' is the first feature selected and added to the bag, then 'Number of NNP' is selected and added aso. Common features are in bold and final results in italics. The plus and the minus signs denote WordNet and collocation relations, respectively.

increases in tandem with the number of tokens labelled as NNP it contains. The Pearson's r is 0.76 (MRR) and 0.89 (Recall), which in both cases signal a strong correlation.

In some sense, our experiments supports the intuition that proper nouns or other named entities in user queries help in finding good answers and ranking them high: A larger number of NNPs cooperates on narrowing the semantic range of a query, because it signals that a query bearing a specific relation. A more specific semantic relation also seems to boost the similarity with a specific answer passage, thus enhancing its MRR score. On the other hand, a more specific relation between entities within the query helps to fetch a larger set of answer passages, since the chances of matching a query entity token increases. In case of search engine queries, the fact that users do not capitalize entities is detrimental as many entities (sequences of NNPs) are unnoticed by an out-of-the-box POS tagger. In our collections, between 2-4% of the paraphrases contain a NNP-labelled token. This sheds some light on the obstacles to reach the upper bound of performance.

Additionally, the greedy algorithm enhanced the performance by augmenting the attribute sets with 19 (Recall) and 18 (MRR) elements (see table 2), which yields 0.69% better Recall and 0.92% better MRR. Nine of this extra features are contained in both lists. Especially, the maximum value of the fellegi-sunter string distance helped to identify effective paraphrases that aimed at juxtaposing or establishing relations among objects, where these objects bear some similarities in their names (Cohen, Ravikumar, and Fienberg 2003).

Ranker 1 (best performance)	Ranker 2 (worst performance)
BOW	BOW
Number of NNP	Bigrams
Number of JJR	Number of NNP
felligi-sunter	**Number of VBG**
Bigrams	preposition^{-}
Word Lemma	**Number of VBZ**
Instance Hypernyms^{+}	**Number of VBP**
Similars^{+}	**adverb**$^{-}$
adverb^{-}	**followingVerb**$^{-}$
Number of SYM	Number of SYM
Number of NNPS	Similars^{+}
Topics^{+}	Number of NNPS
Substance Meronyms^{+}	Number of FW
Number of RBS	**Verb Groups**$^{+}$
Number of UH	Usages^{+}
Number of RBR	Substance Holonyms^{+}
Attributes^{+}	Member Meronyms^{+}
Adjective^{+}	Number of UH
Substance Holonyms^{+}	Attributes^{+}
Entailment^{+}s	Adjective^{+}
Number of RP	Number of RP
	Related^{+}
	Regions Members^{+}
	Trigrams

Table 3: Features acquired for each ranker (Recall). Verb-related features are in boldface.

Thus, the answer is highly likely to contain the same aliases as the query, since the similar part (normally a portion of a stem) is contained in all objects and each suffix indicates an attribute that differentiates each object. The linguistic property of these names seems more beneficially for search than the usage of other longer and more descriptive aliases like full names, e.g.: "*how do i transfer files from my **ps1** mem card to my **ps2** mem card?*", "*how to tell a jeep **4.0** from a jeep **4.2***" and "*combining **pc3000** and **pc400** memory in motherboard*".

Similarly, we check the Recall collection and discovered that the average Recall of a paraphrase systematically increases in when the number of terms labelled as comparative adjectives (JJR) contained in the paraphrases also increases. The Pearson's $r = 0.65$ signals a strong correlation. Overall, 15 fine-grained semantic relations provided by WordNet were useful, but with limited impact.

Error Analysis For a more in-depth error analysis, we split each collection into two data-sets using the error obtained by SVMRanker(G): the first split contains the half of the rankings, for which SVMRanker(G) obtained the best performance, and the second split includes the half, for which SVMRanker(G) achieved the worst performance. Error is defined here as the difference between the best and the achieved MRR/Recall. Then, two rankers, one for each metric, were trained by utilizing the same attributes and learning to rank approach (SVMRank), five-fold cross-validation, and the same feature selection algorithm than SVMRanker(G).

Table 3 compares the attributes acquired for each Ranker when dealing with the Recall collection. Interestingly

Question Type	Number of Samples	Upper Bound	Titles	Lower Bound	BoW (G)	SVM Ranker (G)	BoW (QS)	SVM Ranker (QS)
				MRR				
Why	6853	0.393	0.110	0.0002	0.0828	0.0914	0.0775	**0.0923**
Who	1953	0.437	0.126	0.0003	0.1020	0.1161	0.0949	**0.1195**
Where-info	2142	0.467	0.121	0.0008	0.1055	0.1180	0.1018	**0.1293**
Where-general	5903	0.431	0.125	0.0003	0.1026	**0.1127**	0.1004	0.1123
When	4668	0.402	0.111	0.0003	0.0847	0.0933	0.0797	**0.0963**
What-misc	9967	0.410	0.113	0.0002	0.0881	**0.0960**	0.0844	0.0948
What-do	5648	0.414	0.121	0.0003	0.0972	**0.1059**	0.0938	0.1045
How-extent	8003	0.427	0.120	0.0003	0.0939	0.1016	0.0915	**0.1030**
How-proc	22055	0.406	0.114	0.0003	0.0909	0.0971	0.0899	**0.0981**
				Recall				
Why	8640	0.2372	0.143	0.0039	0.1130	0.1208	0.1126	**0.1212**
Who	3061	0.365	0.211	0.0135	0.1852	0.1992	0.1889	**0.2038**
Where-general	2977	0.331	0.174	0.0038	0.1501	0.1600	0.1490	**0.1633**
When	4673	0.259	0.146	0.0035	0.1249	**0.1300**	0.1221	0.1278
What-misc	9775	0.301	0.156	0.0039	0.1405	**0.1443**	0.1386	**0.1443**
What-do	5482	0.272	0.144	0.0033	0.1360	0.1376	0.1313	**0.1382**
How-extent	6101	0.3045	0.169	0.0038	0.1406	0.1490	0.1410	**0.1495**
How-proc	14237	0.293	0.165	0.0032	0.1423	**0.1479**	0.1411	0.1474

Table 4: Summary of performance per question-type. Only prominent kinds for each data-set are considered.

enough, we can notice that a) Verb-related features are dominant in Ranker 2 (e.g., the following verb and the number of VBG), and they do not appear neither in Ranker 1 nor SVM-Ranker(G) (cf. table 2); b) Only half of the features selected for Ranker 2 are also included in the set of Ranker 1; c) 19 out of the 21 attributes acquired by Ranker 1 are also on the list obtained by SVMRanker(G) (cf. table 2); and d) 10 out of the 24 features in Ranker 2 are also in SVMRanker(G) (cf. table 2). All things considered, we can conclude that SVMRanker(G) fails to correctly rate paraphrases when the discriminative information is mainly verb-related.

Models Based on Question Types Table 4 shows the results obtained for traditional classes of questions. We identified each kind of question by checking whether any of the paraphrases of each ranking starts with any of the patterns provided by an external taxonomy, cf. (Hovy et al. 2000). For each subset, we recompute the bounds and the performance reaped by the question titles, and the performance of the general models. The figures indicate that the improvements achieved by SVMRanker(G) wrt. our baseline BoW(G) are quite consistent in all types of questions in terms of MRR and Recall, attributing in both cases the greatest growth to Who-questions (ca. 13.8% and 7.5%).

The upper bounds obtained for Recall indicate that rankings aimed at Who-questions contain a substantially more effective top-scored paraphrase than When-questions. It seems to be harder to generate/find efficient paraphrases (in terms of Recall) for When-questions. We interpret this as a consequence of the few context that these answers might bear. More specifically, for this sort of questions, some answers only provide the date without context, thus making it difficult to fetch all answer strings, and to accomplish a high Recall. We also trained two question specific learning to rank models for each data-set: one taking into account the BoW property only, and the other by running our feature selection algorithm. These two new question-type specific models are denoted by BoW(QS) and SVM-Ranker(QS), respectively. All the figures were obtained by performing five-fold cross validation. Results indicate that our features yielded an improvement in performance for all types and questions and data-sets. The greater improvements in Recall and MRR were due to where-general (9.6%) and where-info (27%) types of questions, respectively. Only in five out of seventeen cases the performance was detrimental wrt. the general models. Hence, we conclude that question-type models are a more effective alternative to general all data-encompassing ranking models.

Conclusions and Future Directions

We presented a novel method for rating query paraphrases for effective search in cQA. It exploits query logs from a commercial search engine and Yahoo! Answers for collecting a corpus of paraphrases. These paraphrases are automatically annotated in terms of recall and MRR, and thus used for building several learning to rank models, i.e. general and specific to question-types, and according to both metrics.

In a nutshell, our results indicate that question-type models are more effective than general ranking models. They also point out to the need for good-performing NLP toolkits for queries. Research into this area is still incipient. Our error analysis suggests that the idea of ensembling rankers of different nature sounds promising as it can provide a comittee of leading experts, each focused on a particular class of ranking. Here, we envisage the use of clustering algorithms to find these (not yet defined) number of experts, and the devise of a specialised ensembler that chooses the expert in agreement with the characteristics of the problem.

Acknowledgements

This work was partially funded by the project FONDEF-IdeA (CA12I10081) funded by the Chilean Government.

References

Cohen, W. W.; Ravikumar, P.; and Fienberg, S. E. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. In *International Joint Conference on Artificial Intelligence*, 73–78.

Hovy, E. H.; Gerber, L.; Hermjakob, U.; Junk, M.; and Lin, C.-Y. 2000. Question answering in webclopedia. In *TREC*.

Ji, S.; Zhou, K.; Liao, C.; Zheng, Z.; Xue, G.-R.; Chapelle, O.; Sun, G.; and Zha, H. 2009. Global ranking by exploiting user clicks. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 35–42.

Joachims, T. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Lin, C.-Y. 2008. Automatic question generation from queries. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge*, 929–937.

Radlinski, F.; Szummer, M.; and Craswell, N. 2010. Inferring query intent from reformulations and clicks. In *WWW 2010, Proceedings of the 19th Annual International World Wide Web Conference*.

Surdeanu, M.; Ciaramita, M.; and Zaragoza, H. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics* 37(2):351–383.

Suzuki, S.; Nakayama, S.; and Joho, H. 2011. Formulating effective questions for community-based question answering. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*, 1261–1262.

Wen, J.-R.; Nie, J.-Y.; and Zhang, H. 2002. Query clustering using user logs. *ACM Trans. Inf. Syst.* 20(1):59–81.

Xu, J.; Chen, C.; Xu, G.; Li, H.; and Abib, E. R. T. 2010. Improving quality of training data for learning to rank using click-through data. In *WSDM'10 , Proceedings of the Third International Conference on Web Search and Web Data Mining*, 171–180.

Zhao, S.; Wang, H.; Li, C.; Liu, T.; and Guan, Y. 2011. Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 929–937.

Zhao, S.; Wang, H.; and Liu, T. 2010. Paraphrasing with search engine query logs. In *COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, 1317–1325.

Zhao, S.; Zhou, M.; and Liu, T. 2007. Learning question paraphrases for qa from encarta logs. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1795–1801.

Zheng, Z.; Si, X.; Chang, E.; and Zhu, X. 2011. K2q: Generating natural language questions from keywords with user refinements. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 947–955.