From Query to Question in One Click: Suggesting Synthetic Questions to Searchers

Gideon Dror, Yoelle Maarek, Avihai Mejer, Idan Szpektor
Yahoo! Research
Haifa 31905, Israel
{gideondr,amejer,idan}@yahoo-inc.com, yoelle@ymail.com

ABSTRACT

In Web search, users may remain unsatisfied for several reasons: the search engine may not be effective enough or the query might not reflect their intent. Years of research focused on providing the best user experience for the data available to the search engine. However, little has been done to address the cases in which relevant content for the specific user need has not been posted on the Web yet. One obvious solution is to directly ask other users to generate the missing content using Community Question Answering services such as Yahoo! Answers or Baidu Zhidao. However, formulating a full-fledged question *after* having issued a query requires some effort. Some previous work proposed to automatically generate natural language questions from a given query, but not for scenarios in which a searcher is presented with a list of questions to choose from. We propose here to generate synthetic questions that can actually be clicked by the searcher so as to be directly posted as questions on a Community Question Answering service. This imposes new constraints, as questions will be actually shown to searchers, who will not appreciate an awkward style or redundancy. To this end, we introduce a learning-based approach that improves not only the relevance of the suggested questions to the original query, but also their grammatical correctness. In addition, since queries are often underspecified and ambiguous, we put a special emphasis on increasing the diversity of suggestions via a novel diversification mechanism. We conducted several experiments to evaluate our approach by comparing it to prior work. The experiments show that our algorithm improves question quality by 14% over prior work and that adding diversification reduced redundancy by 55%.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: *Question-answering systems*

General Terms

Algorithms, Experimentation

Keywords

Community-based Question Answering, Question generation

1. INTRODUCTION

Web search engines fail to satisfy users' queries in three cases, either the search engine is faulty (it did not crawl, index, retrieve or rank the relevant content adequately), or users did not express their intent adequately (hence the need for query assistance tools) or, finally, relevant content for the specific intent does not exist on the Web at the time the query is issued. We are focusing here on this latter case, which can occur for multiple reasons: the user's need may be rare, narrow, complex, heterogeneous, etc., or the Web content in that specific country/language is still scarce. In such cases, a common solution is for users to turn to Community Question Answering (CQA) sites, such as Yahoo! Answers, Baidu Zhidao or specialized forums like StackOverflow, and ask for other users to generate the missing content. To do so, they need to switch sites and invest some effort in expressing their intent as a question that is intended to be read by human beings. Ironically, users are so used to issue a few-word queries by now that expressing a natural language question might be perceived as tedious. Some of the authors of this paper have studied the transition of "searchers" into "askers" and discovered that even within users of CQA sites, only 2% take the pain to do the transition [16].

As a step forward in facilitating the transition from searchers to askers, we propose here to suggest to a frustrated searcher[1] a few synthetic natural-language questions that were automatically generated from the searcher's query. Searchers can then post a question in one click (or after minor editing). Our general approach follows prior work in this area: offline learning of question templates, such as '*where should I T1 a T2 in T3?*', and instantiating them online, with words from the issued query [25, 24]. This approach allows to automatically generate original relevant questions, which may have never been asked before by users. However, these prior studies did not consider the scenario in which a searcher views the generated questions and is expected to directly click on them to post them. Particularly, on top of generating questions that are relevant to the query, our target scenario imposes specific requirements on the quality of the generated questions. First, the displayed questions have to be grammatically correct and natural enough to be posted by human beings. Using only basic language models would lead to suggestions that should not be shown to users, such as "*where rent villa in italy?*"[2] for the query "*italy rent villa*", instead of the more appropriate "*where should I rent a villa in italy?*".

In addition, we argue here that evaluating the generated questions on an individual basis is fine when these questions are used for behind-the-scenes improvements, such as query rewriting or ranking [24]. Yet in our scenario, like in search results, another major requirement is that the list of displayed questions do not include near-duplicates. Furthermore, it should be as diverse as possible,

[1]We note that detecting queries with question intent or identifying frustrated searchers is out of the scope of this paper.

[2]Such suggestions were generated by our implementation of [24], which we describe later.

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

while still relevant to the original query. Indeed, given that queries tend to be under-specified, a diverse set of suggested questions is more likely to include a candidate that catches the actual intent of the user. More generally, a diverse list of suggestions is probably perceived as more attractive, and would hopefully tease users into clicking on one of them.

Given the above requirements, we propose a query-to-question recommendation approach based on a learning-to-rank framework. Under this framework we employ a large set of features that address aspects of question relevance to the original query as well as aspects of grammatical correctness. These features capture more complex context and syntax mismatches, such as predicate argument selection (*e.g.* that '*fix car*' is more plausible than '*fix cat*'). In addition, to promote diversity in the suggested list, we introduce a novel diversification component that filters out redundant variations of the same question. We argue that the notion of similar questions goes beyond traditional paraphrasing, as "*what is a really good way to lose weight?*" and "*what is your favorite way to lose weight?*" share the same question intent. We thus extract and utilize interchangeable terms such as *good/favorite* via an edit distance algorithm in order to identify such question similarities.

We compared our algorithm to an implementation of Zhao et al. [24] as baseline under several experiments: manual experiments, in which the results are evaluated by professional human judges, and large-scale automatic experiments on 150,000 gold-standard query/question pairs. The results of the manual evaluations show that both the prior work baseline and our system achieve very high relevance scores, but our algorithm significantly performs better in terms of grammatical correctness. In addition, the novel diversification component dramatically reduces the number of redundant questions, with only a modest decline in question quality. Our large-scale experiments show that our algorithm achieves significantly higher recall, ranking higher the specific questions chosen by real users for their queries. Furthermore, when diversifying the suggestions, a better coverage of question intent is reached, further increasing recall.

The main contributions of this paper are threefold. First, we propose a general learning framework for ranking question suggestions that leverages a large set of features, with special attention given not only to relevance but also to the grammatical correctness of the synthetic questions. Second, we introduce the concept of diversification in question suggestions to improve the overall quality of the suggested question list. Finally, we use a novel large-scale automatic evaluation mechanism that shows that our system performs better in terms of both relevance and correctness, as compared to previous work. We believe that these contributions are critical to make the question suggestions list attractive enough to be shown to users and eventually clicked on.

2. TEMPLATE EXTRACTION

Our general approach for automatically generating recommended questions for a given query is conducted in two steps. In a first stage, we extract question templates from a very large dataset of query/question pairs. Then, in a second stage , we instantiate question templates with the words of a new query so as to generate synthetic questions, and then rank the latter. In this section, we discuss the first stage, namely data gathering and the extraction of question templates.

In order to generate question templates, it is most useful to observe the relations between queries and questions with similar intent. Some of the authors of this paper have generated such datasets[3] of query/question pairs in the past, in which the query is issued on a search engine, and the question was a search result (coming from a CQA site) on which the searcher clicked [15]. The innovative step taken by Zhao et al. [24] was to extract question templates from such datasets, which they built over Baidu's queries and Baidu Zhidao's questions. Interestingly, Zheng et al.[25] used a similar dataset to evaluate their question generation process, but not to generate templates.

(query, question)	question template
("*love Justin Bieber*", "*Why do people **love Justin Bieber** so much?*")	Why do people **T1 T2 T3** so much?
("*best ever US basketball player*", "*Who is the **best ever basketball player** in the **US**?*")	Who is the **T1 T2 T4 T5** in the **T3**?
("*movies downloading free*", "*What are the best sites for **downloading movies for free**?*")	What are the best sites for **T2 T1** for **T3**?

Table 1: From (query,question) pairs to question templates

We follow here the approach of Zhao et al. and apply their template extraction algorithm on a query/question dataset, modulo some slight modifications. First, we built a dataset consisting of 28 million pairs of (query,question), in which the question originated from Yahoo! Answers, keeping only the title as the question's text. We retained only pairs in which all query terms appear in the question, as an easy approximation of ensuring relevance to the query. Differing from Zhao et al., who focused only on short Chinese queries containing 1-3 terms, we considered English queries consisting of 3-5 terms. This choice was driven by previous studies, such as [12], that showed that shorter queries were rarely expressing an information need, let alone a question intent. We did not go either towards longer queries counting more than 6 terms, as these become too specified to be good candidates for abstracting templates. Finally, to ensure the quality of the extracted templates, we used a simple heuristic, and kept only questions that start with one word that belong to a manually-defined white list. This white list holds about 20 words, such as the so-called six "WH question words" (*what*, *who*, etc. and *how*), as well as some auxiliary verbs (*e.g.*, *should*, *can*, *is*, etc.) This heuristics plays the role of a very simplistic shallow parser and ensures the elimination of questions such as "*Photosynthesis question?*" or "*arrested for less than a gram of marijuana (youth)?*". Following this filtering, we ended up with 4.2 million query/question pairs.

From the 4.2 million pairs in our dataset, templates are extracted by substituting the query terms found in the question with "slot fillers", exactly in the same manner as done by Zhao et al. For example, from the pair formed by the query, '*fix old car*', and the clicked question "*how can I **fix** my **old car**?*", we extracted the template "*how can I **T1** my **T2 T3**?*". We encode the query term position as part of the variable name, to avoid mistakes induced by asymmetric relations for instance. Thus, "*Is **T1** taller than **T2**?*" and "*Is **T2** taller than **T1**?*" are considered different templates. See Table 1 for additional examples[4]. We then filter out rare templates,

[3]More sophisticated yet smaller datasets were also generated, in which the question is not a search result, but a question generated and posted by a searcher in a same session that started by issuing the query in the pair, [16].

[4]Note that these have been slightly modified for privacy reasons, in order to avoid linking specific public Yahoo! Answers askers to their search queries.

keeping only templates that are associated with at least 10 different queries. At the end of this process we generated a database of about $40,000$ templates together with their associated queries.

3. AUTOMATIC QUESTION SUGGESTION

We described in the previous section how the question template database is constructed. We now explain the second step of our process, how given a new query we generate and rank template-based questions.

The two previous works in question generation from queries [24, 25] proposed each their own scoring functions for ranking candidates. Given the complexity of the task, we preferred here using a learning-to-rank approach. This enables us to employ a much larger set of features that capture different aspects of the quality of each question.

Our approach requires to first generate all possible candidate questions for a given input query, then represent each generated question as a rich feature vector, and finally, rank them using a discriminative model. Since the number of candidates can be very large (in the order of thousands) and our features are computationally expensive to generate, we propose to first generate a smaller pool of candidates, generate the richer features for these candidates, and only afterwards apply our learning-to-rank mechanism on this pool. This type of two-step ranking has often been applied in Information Retrieval in the past [2, 5], as well as in natural language generation tasks [10, 13]. In order to generate this pool, we first rank questions using a reimplementation of Zhao et al. [24] algorithm, which also serves as our baseline in our experiments. We refer to this as the *baseline step*. Our reranking algorithm is then applied on the top-K candidates of the baseline output. We refer to this second step as the *reranking step*.

We next detail the different parts of our algorithm. We first describe how question candidates are generated for a given query. We then present the different features induced from each question, followed by a description of our learning-to-rank model.

3.1 Generating Question Candidates

The first part of the baseline step consists of generating all the candidate questions for a given query. To this end, each template in our template database is considered, and templates that are found relevant are instantiated with words from the query.

As we intend to compare ourselves to Zhao et al. as baseline, we use the same methodology in order to select relevant templates. Namely, a template is considered relevant to the query only if the query is similar enough to one of the queries associated with the template in the database. Specifically, two queries are considered similar if they count the same number of terms and share at least one term in the same query position. For example, the queries "*fix* ***old*** *car*" and "*buy* ***old*** *records*" are considered similar since both have three terms and share the word '*old*' in the second position. On the other hand, "*fix* ***old*** *car*" and "***old*** *car repair*" are not considered similar, even though they share two words, since they have no shared word in the same position.

We identify similar queries in an efficient manner by indexing all queries in the template database by their words and positions of each word in the query. Thus, given a new query q, we fetch all similar previous queries from the template database to produce a set $\{q_i\}$. Then, for each q_i, we fetch the templates t_j that were associated with it in the template database, and instantiate each t_j with the words of q. This process produces as many candidate synthetic questions as the t_j's. We emphasize that the instantiation of templates with new queries may result in candidate questions that may have never been asked before by users.

3.2 Question Feature Representation

To rank the different question candidates, for both completing our baseline step as well as conducting our reranking, we transform each question into a feature vector. Each feature captures various semantic and syntactic aspects of the question as described next.

3.2.1 Baseline Features

The first two features we generate for each question are the ones suggested by Zhao et al. [24]. These features are also the only features we use in the baseline step to generate a pool of candidate questions.

The first feature is the likelihood score for the query to instantiate the template behind the generated question, referred to as $likelihood$. It is computed for a pair of query/template (q_i, t_j) by averaging the similarity scores $sim(q, q_i)$ between the new query q and the n similar queries q_i, that are associated with template t_j. Each similarity score is computed as the product of the term similarity between terms in the same position in the two queries, *i.e.* $sim(q, q_i) = \prod_k sim(q^k, q_i^k)$, where q^k and q_i^k are the k^{th} terms of queries q and q_i respectively. Finally, each term is associated with a context vector that includes all the terms that co-occur with it in the query set, weighted by tf·idf. The similarity $sim(q^k, q_i^k)$ between two terms is measured by the cosine similarity between their context vectors. In our specific dataset, we used a sampled Yahoo! query log consisting of 18 million queries to compute the context vectors of all terms.

The second baseline feature is a trigram language model score for the question. This feature reflects the grammatical correctness of the question. In our implementation, we trained this language model on a random sample of 15 million English questions from Yahoo! Answers using the *berkeleylm*[5] package [21]. We only considered questions that start with one of the words in our question white list, as done in the template extraction stage (see Section 2).

These two baseline features (and only those) are used in the baseline step to provide a baseline score for each question, as per the following linear combination:

$$\begin{aligned} baseline\text{–}score = \; & \lambda \cdot likelihood(Query, Template) \\ & +(1-\lambda) \cdot LanguageModel(Question) \end{aligned}$$

Zhao et al. learn only a single parameter, λ, in order to tune the influence of the two features. This concludes the baseline step that generates a pool of K candidates that achieved the highest $baseline\text{–}score$ values. We detail below the additional features that are needed for proceeding to the reranking step.

3.2.2 Reranking Features

We parse each candidate question at two granularity levels: each word is tagged with a part of speech (POS) tag, and the dependency parse tree of the question is generated. For both POS tagging and dependency parsing, we use the Stanford NLP tools[6] [8]. We next describe the reranking features extracted per question, which are mainly extracted from this syntactic parse information.

Question POS language models.

We compute the five-gram language model scores for the question POS tags and for the coarse-POS[7] (CPOS) tags. This provides a high level syntactic fluency assessment of the question. The POS

[5] `http://code.google.com/p/berkeleylm/`

[6] `http://nlp.stanford.edu/software/`

[7] The Coarse-POS tag consists of only the first letter of the standard POS tag, such as N for nouns, without distinguishing between singular/plural and common/proper nouns (NN, NNS, NP, NPS).

and CPOS language models were trained on the POS tags for the same 15 million question dataset used for training the trigram language model above.

Query POS sequences.

We tag the query words with POS by copying the tags that were assigned to the query words in the question, which was built as an instantiated template. For example, for the query "*fix old car*", the candidate question "*should I **fix** my **old car**?*" induces the query POS tags "*VB JJ NN*", while the candidate question "*how much is a **fix** for an **old car**?*" induces the query POS tags "*NN JJ NN*".

Given the induced POS and CPOS tagging of a query, we generate binary features representing all the position dependent unigram, bi-gram and tri-gram subsequences within the query tag sequence. For example, for the POS tag sequence "*VB JJ NN*" we generate the features '*1-VB*', '*2-JJ*', '*3-NN*', '*1-VB-JJ*', '*2-JJ-NN*', and '*1-VB-JJ-NN*'. Overall, there are about 39,000 such possible features, which capture the a-priori tendency of queries to map into specific tag sequences, and attest to the validity of placing them in the corresponding template slots.

Dependency Relations.

We generate binary features that correspond to the lexical and to the POS/CPOS head-dependent pairs that appear in the top level of the parse tree. As an example, the parse tree for the question "*should I fix my old car?*" is shown in Fig. 1. The head-dependent binary features '*should* $\rightarrow$ *fix*', '*I* $\rightarrow$ *fix*', '*car* $\rightarrow$ *fix*', '*MD* $\rightarrow$ *VB*', '*PRP* $\rightarrow$ *VB*', '*NN* $\rightarrow$ *VB*', '*M* $\rightarrow$ *V*', '*P* $\rightarrow$ *V*' and '*N* $\rightarrow$ *V*' are derived from this parse tree. These features capture semantic coherence such as verb selection preferences. The strength of these features lies in their ability to capture relations between words that are too far away to be assessed using a standard n-gram language model. We focus only on the relationships between the root word and its dependents, as we empirically noticed that the validity of the main theme of the question, as captured by the top elements of the parse tree, reflects the overall quality of the question, especially since our questions are usually quite short. There are about 2.6 million such features in our training set.

Parse Tree Score.

An additional feature we consider here is the score provided by the parser to the generated parse tree. Lower scores indicate less probable parse tree. The parse tree score has been previously shown as useful for reranking in natural language generation [10, 13].

Template Word Order.

Independently of the parse information, we also extract a binary feature that indicates the slot order in the template that generated the question. For example, for the template "*should I **T4** my **T1 T2 T3**?*", we generate the feature '*T4-T1-T2-T3*'. This feature captures a prior indication of whether a specific word order is likely to be useful for a query. For example, '*T1-T2-T3-T4*' is more likely to be useful than '*T4-T3-T2-T1*'.

3.3 Learning-to-Rank model

We detail below the learning-to-rank model that we apply on the pool of K candidate questions generated for each query during the baseline step. In this step, each candidate question is represented by a vector containing all the features described in Section 3.2. We use a linear model for scoring each question vector in the candidate pool: $\mu \cdot \mathbf{\Phi}(Q)$, where μ is the model weight vector and $\mathbf{\Phi}(Q)$ is the feature vector of question Q. The weights are trained using the averaged variant of the Passive-Aggressive (PA) online learning

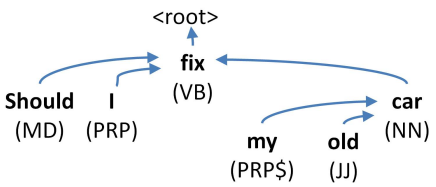

Figure 1: POS tagging and dependency parse tree for the question "*should I fix my old car?*"

algorithm [6], which showed competitive results for large feature spaces [18, 7]. This learning procedure is summarized in Alg. 1. At each step, a training pair[8] consisting of a query and a corresponding clicked question, which we denote the *target question*, is considered for updating the current model μ_t. Given the K question candidates for the query in the training pair, if the target question is not found among them, the current pair is skipped. Otherwise, the full feature vectors are computed for all K candidates, which are then scored according to the current reranking model μ_t. At this stage, if the scores of some questions are not lower by a sufficient margin from the target question's score, the model weights are updated accordingly to promote the score of the target question with respect to these other questions, resulting in a new model μ_{t+1}. The algorithm iterates T times over the entire training set of M examples and the average of the weights at each step in the algorithm are set as the final model.

For training and evaluating our ranking model, we use a subset of our query/question pairs dataset generated in Section 2.This subset consists of queries that are associated with only a single template among the set of templates we extracted, resulting in about 1.6 million pairs. Thus, in each training step, it is guaranteed that only one question out of all candidate questions is considered the "correct" one. This is the question that was generated from the single associated template above. This training set allows us to tune the model towards real user choices of questions related to their queries. We split the subset of queries to training, development and testing sets, respectively as 80%, 10% and 10% of the data. We refer to this data set as our gold standard.

One important point to note here is that this subset of the data that we use consists of the query/question pairs that were also used to populate the template database. So, we could have easily selected the correct target template for each of the queries. To avoid this, when processing a specific query, we ignore this query in the database. Specifically, in the stage of matching a query to other similar queries in the database in order to collect candidate templates, we do not match a query to itself. We can thus simulate the situation in which every query is a new unseen query.

4. RANKING EVALUATION

4.1 Experimental Setup

To evaluate our reranking model, we compared it to our reimplementation of Zhao et al. algorithm [24], which serves as baseline, via two experiments: a manual and a large-scale automated one, as detailed below.

Several parameters need to be tuned both for the baseline and reranking steps. Parameter tuning was conducted using the development dataset we put aside (see Section 3.3). We set the value of λ, which tunes the two features used in the scoring function in

[8]The extraction of these pairs is detailed in Section 2.

Algorithm 1 Ranking model PA training

Input:

- M training examples, each with K question candidates
- N number of updates at each round
- Feature mapping function $\mathbf{\Phi}(Q) \in \mathbb{R}^d$

Initialize:

- $\boldsymbol{\mu}_0 = \mathbf{0}$

For $t = 1, 2, \ldots, M$

- Get question candidates for query q_t: $\{Q_1, \ldots, Q_K\}$
- Get correct question Q_j
- Rank all candidates by $\boldsymbol{\mu}_{t-1} \cdot \mathbf{\Phi}(Q_i)$, $i = 1, 2, \ldots, K$
- **For** $i = 1, 2, \ldots, N$ top candidates
  - Define $\Delta_{j,i} = \mathbf{\Phi}(Q_j) - \mathbf{\Phi}(Q_i)$
  - Compute

$$\alpha = \frac{\max\left\{0, 1 - \boldsymbol{\mu}_{t-1} \cdot \Delta_{j,i}\right\}}{\|\Delta_{j,i}\|^2}$$

  - Set: $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \alpha \Delta_{j,i}$

Output: Averaged weight vector $\mu = \frac{1}{M} \sum_{t=1}^{M} \mu_t$

the baseline step, to 0.2. This value maximized the Mean Reciprocal Rank (MRR) score for the known target question of each query in the development set. Note that setting this value results in giving some extra weight to the language model score in the formula. This value is slightly different from the value that was assigned by Zhao et al. in their original algorithm, but we believe that using exactly the same value, rather than the one that performs best on our data, would have been unfair to the baseline, as we are dealing with different languages.

The second parameter to tune is K, the number of top question candidates to keep from the baseline step as the pool of candidate questions to be reranked. There is a clear trade-off between effectiveness, which will increase with recall when there are more candidates, and efficiency, which decreases when more candidates need to be processed. We define as *recall@K* the percentage of queries, for which the known target (or gold-standard) question is ranked among the top K candidates. Table 2 shows the recall@K values obtained on the development set for different values of K. As per the table, we decided to set $K = 100$, as a good balance between the processing load and the probability of having the target question among the top candidates.

Finally, we tuned the hyper-parameters of the ranking model training procedure, specifically T, the number of training iterations, N, the number of updates at each round, and whether or not to use parameter averaging. We chose $T = 3$, $N = 5$ and decided to use parameter averaging, as it maximized the MRR score with respect to the target questions in the development set.

4.2 Manual Evaluation

For the manual evaluation, we randomly sampled 1,000 queries from the prepared test dataset (see Sec. 3.3). For each query we collected the top three question suggestions as ranked by the baseline and the reranking steps. We limited ourselves to the 3 best suggestions as this seems to be a reasonable size of a list to be shown to users, be it on the search result page or on the CQA site itself. We asked professional judges, also known as *editors*[9], to assess each question by two metrics: (a) *relevance* – which reflects whether the question's content is reasonable and relevant to the query (ignoring grammar mistakes); (b) *grammar* – which reflect the grammatical correctness, our second requirement. The editors where asked to provide a binary *good/poor* score for to each metric. Table 3 shows an example question for each of the possible assessment combinations of the query "italy rent villa". Three human editors evaluated the test set, and the final assessment for each question suggestion was set to be the majority choice. While we separately evaluated the relevance and grammatical correctness of each algorithm, a question cannot be shown if it does not excel in both. Therefore, we also computed a *combined* metric, which is 'good' only if both *grammar* and *relevance* are 'good'.

[9]These judges are members of Yahoo! "editorial team" and have a long experience of assessing relevance, and quality in general, across multiple Yahoo! products.

K	25	50	100	200
Recall@K	76%	84%	91%	93%

Table 2: Recall@K for several values of K in development set

Question	Relevance	Grammar
where can i rent a villa in italy?	good	good
where to rent villa in italy?	good	poor
why can you rent a villa in italy?	poor	good
is italy rent a villa?	poor	poor

Table 3: Examples of questions for the query "italy rent villa" together with their relevance and grammar scores

Table 4 presents the results of the manual evaluation. The numbers represent the percentage of questions that were assessed as 'good' under each metric. With respect to relevance, both algorithms show very high performance, as 94-96% of the top 3 suggestions were found relevant by the judges. We think that this is due to the fact that all query words appear in each of the questions, and thus most questions seem somewhat relevant. It is thus not surprising that the reranking algorithm did not improve over the baseline, as there was not much to improve on from the start.

Compared to relevance, the grammar metric results are much lower. Only 70% of the highest ranking questions generated by the baseline were considered grammatically correct by the editors, and this value decreased to 60% for the suggestions in the second and third ranks. It is here that our reranking algorithm showed its benefit, achieving an increase of 14% in correct questions for the best suggestion and an increase of 10% for the second best suggestion. These improvement ratios remain stable for the combined metric, since it is almost a duplicate of the grammar metric. We note that Zhao et al. [24] also conducted a manual evaluation for Chinese generated questions, which considered both the relevance and the fluency of the question. Under their evaluation, the baseline algorithm achieved a score of 67%, which is not far from the 71% it achieved under the *combined* metric, in our manual evaluation conducted by different people on a different language/dataset.

4.2.1 Error Analysis

As evident from the manual evaluation, most incorrect questions are due to grammatical mistakes. While the reranking algorithm reduced the number of grammatically invalid questions by more than 30%, still about 19% of the top suggestions contain grammatical mistakes. Examples of such mistakes are shown in Table 5. The most common mistake is the misuse of the singular/plural form that leads to the incorrect selection of *is/are* with *a/some*, and less frequently to a conjugation mistake, such as *do/does*. Another rel-

	Baseline			Rerank		
	relev.	gram.	comb.	relev.	gram.	comb.
Q_1	96.6%	71.3%	71.2%	96.4%	81.5%	81.2%
Q_2	96.0%	60.1%	59.9%	95.9%	66.4%	66.1%
Q_3	94.2%	59.6%	58.9%	94.3%	59.5%	59.1%

Table 4: Manual evaluation baseline vs reranking: Relevance, grammar and combined results, for the top 3 questions, $Q_{1,2,3}$, expressed as the percentage of questions that received a "good" score

Question	Mistake type
what are some **good party music**	Singular/Plural
how do the **world get smaller**	Do/Does
what does the **free chat lines**	Be/Do
what **a good nickname for me**	Missing a verb
what is the **french word for** the **bread**	Determiner
who is the **best mp3 player** today what is the **best basketball player** today	Subject type
what was the **force of gravity**?	Question tense
what is the correct **way to die in** an **accident**	Sentiment

Table 5: Examples of incorrect questions, with query terms highlighted in bold

atively common mistake type involves the incorrect usage of determiners, such as a mixup between *a/the*, missing a determiner, or the superfluous addition of an article. It may be possible to design additional features to target some of these specific cases.

The rare cases of relevance mistakes are more subtle and harder to mitigate. One type of such mistakes is a mismatch between the question type and the subject of the question, such as the selection of *who* vs. *what* for questions addressing '*mp3 player*' vs. '*basketball player*'. A second type of mistakes relates to question tense. While it is fine to use the past tense for a question such as "*what was the trigger for ww2?*", it is inappropriate for the question "*what was the force of gravity?*". Finally, some mistakes rise due to sentiment polarization, such as in the question "*what is the correct way to die in an accident?*", where the qualifier *correct* might be seen as awkward or cynical.

4.3 Automatic Evaluation

While the manual evaluation we conducted provides valuable insights, it is nevertheless limited. First, like most manual evaluations, it is of small scale, using only 1,000 questions. Second, estimating the relevance of a question suggestion can only be an approximation, without evidence of the actual query intent. Indeed, for our editors, almost every question that contained the query words seemed to be *a priori* relevant, yet we had no guarantee that they were indeed reflecting the user's implicit needs. We therefore conducted a complementary large scale automatic evaluation over the query/question pair test set. For this purpose, we leveraged the gold-standard dataset defined in Section 3.3, in which a query/question pair, (q, Q), is built by pairing the unique gold-standard question Q that the user clicked on, after issuing the query q. For an algorithm to perform best, question Q should be ranked first in the list of suggested questions for query q. While this evaluation cannot provide an absolute quality assessment for the suggestions, we believe it provides a better indicator of which algorithm suggests questions that real users could have clicked on.

Method	Avg. rank (lower better)	MRR (higher better)	Recall @1	Recall @3
Baseline	8.98	0.54	42%	62%
Rerank	4.45	0.65	52%	73%

Table 6: Performance of rerank vs baseline on several metrics derived from the rank of the gold-standard question

The test set we use counts 160,000 queries. However, we chose to generate pools of 100 candidates at the baseline step, for a recall of 91%, as per Table 2. Therefore, we only use 147,000 queries from this dataset (namely 91% of 160,000) in our experiment. Indeed, only these queries have their associated unique target questions appear in the candidate pools generated by the baseline component. Table 6 compares the performance of our reranking model to the baseline, for various measures, over this gold-standard test set. The rerank algorithm shows a substantial improvement in all metrics compared to the baseline. Namely, the average rank of the target question is halved, from 8.98 to 4.45. In addition, we observe an improvement of recall@1 of 24% and an improvement of recall@3 of 17%, as compared to the baseline. In other words, the reranking algorithm is better at promoting the question with the specific intent of the user at query time.

4.3.1 Features Analysis

We employed the automatic evaluation framework to analyze the contribution of each of the features used for the reranking step. We evaluated both the contribution of each feature family on its own (on top of the baseline features), and its contribution on top of all the other reranking features (via an ablation test). Table 7 summarizes the feature families used, the number of features in each family and their impact on the Recall@1 measure.

From the table, we first notice that the dependency-tree features contribute the most. At the top stands the lexical head-dependent feature family, which achieves alone a 15% improvement over the baseline. This is a significant effect, considering that the full reranking algorithm achieved 24% improvement. Compared to this family, which consists of more than a million features, the single feature of the dependency tree score managed to gain 6% on its own over the baseline, and also contributed nicely on top of the rest of the features. This demonstrates the value of examining the parse tree at different granularity levels. Indeed, the family head-dependent POS pairs is shown to capture information that complements the fine-grained lexical and overall views of the dependency tree.

Encoding sequential syntactic information did not contribute much to the reranking step. The POS language model did provide some useful information on top of other features. However, using the patterns generated by POS mapping on query terms actually hurt the reranking performance, hinting that the order of query words does not generally indicate any specific user intent. Finally, while the template type by itself is not useful for improving ranking, it does help a little on top of everything else as a weak prior.

5. DIVERSIFYING SUGGESTIONS

As queries are typically underspecified, they can easily represent different intents. Let us consider our previous query example "*italy rent villa*", it may be associated with very distinct questions, each representing a different intent, *e.g.*, "*where should i rent a villa in italy?*", "*how much does it cost to rent a villa in italy?*" or "*would you recommend to rent a villa in italy?*". We refer to these as diverse intents and argue that questions need to be diversified to al-

Family	Count	Recall@1 Feature Addition	Recall@1 Feature Ablation
POS/CPOS lang. model	2	1.7%	-2.3%
Query POS/CPOS	39K	1.1%	0.5%
Dependency-tree score	1	6%	-3.8%
Dependency-tree lexical pairs	2.6M	15.7%	-5.3%
Dependency-tree POS/CPOS pairs	68K	6.6%	-3.2%
Template word order	242	-1.3%	-0.8%

Table 7: Influence of the feature families used for reranking. The two right-most columns represent the changes (in percents) of Recall@1, due to respectively adding (column Addition) and removing (column Ablation) a single feature family

low for the occurrence of diverse intents. While diversification was shown to be beneficial to traditional recommender systems [26, 23, 9, 4, **?**], to the best of our knowledge, it has not been considered yet in question generation for queries.

To validate the need for diverse questions, we manually inspected dozens of queries and their associated question recommendations. We noticed that the top questions are often simple variations of one another. For example, for the query "*fix old car*", we generate several questions that all carry the same intent, "*how do I fix my old car?*", "*how can I fix an old car?*" and "*how could you fix your old car?*". Interestingly, while these questions are not formally paraphrases, they do represent exactly the same intent, and differ only by some terms being exchanged, such as *I*/*you* and *an*/*my* in the above examples. We also observed variations of qualifiers or verbs that also carry the same intent, such as in "*where can I {find/buy/get} a TV for a good price?*". In general, we found that the top-5 recommended questions represent on average only 2-3 different intents.

Motivated by the above, we integrate diversification into our algorithm as a post-process filtering step. In this step the algorithm scans the ranked recommendation list, and applies one of the two filtering methods described below.

5.1 Imposing a Different Question Form

A very simple heuristic method for promoting diversification is to impose a different question form, represented by a different interrogative word, namely (e.g. *what, when, where, etc.)*. We extend this approach by forcing each question in the suggestion lists to start with a different word, in order to also cover questions that begin with an auxiliary verb '*Should I ...?, Does the ...?*'. If the first word of each question is different, intuitively there is a good chance that the question form, and therefore the underlying intent, differ. Our heuristic works as follows, we examine the suggested questions starting from the top and eliminate a candidate if it starts with the same word as a previous question. For many queries there are only a few top ranked questions that use a different first word. As a result, this method typically has to examine questions far down the list, in order to find enough candidates that pass the test. We refer to this method as the "first-word" filter in the rest of this paper.

can/could	could/should	can/do	a/the
term/word	type/kind	get/download	buy/get
cool/cute	fun/good	ok/normal	possible/legal
u/you	computer/pc	okay/ok	site/website
do/don't	bad/ok	girls/guys	my/your
really/-	exactly/-	often/-	always/-

Table 8: Sample of automatically identified interchangeable terms

5.2 Eliminating Redundant Questions

An additional way to ensure diverse questions is to eliminate redundancy. We consider as redundant, questions that differ by only a few "interchangeable" terms, namely terms that can be exchanged without changing the intent of the question. Assuming we have at our disposal a list of such interchangeable term pairs, we propose to measure the redundancy between questions by a term-based edit distance metric that inflicts nonuniform penalties for different edit operations. An edit operation that involves replacement between two interchangeable terms costs less than other operations. We note that one of the interchangeable element in the pair can be an empty string, thus representing a deletion operation. After some experimentation and manual evaluation, we assigned $cost = 1$ to the low cost operations and $cost = \infty$ to all other operations. We consider two questions to be redundant if the edit distance between them is lower than a threshold K (K=3 in our experiments). We refer to this method as the "edit-distance" filter in the rest of this paper.

In order for this filter to work, we need to provide a list of interchangeable term pairs. We extract such pairs in an automatic way, based on the assumption that terms that appear in the same context in many different questions are interchangeable. To this end, we utilize a large dataset of 1.2 million queries and the top 50 questions generated for each of them by our recommendation algorithm. For each query, we examine the top generated questions and identify among them the pairs of questions that differ by exactly one term. We then record the corresponding term pair in each such occurrence, also allowing for one term to be replaced by a blank in the case of a deletion operation. We finally consider two terms as interchangeable if the pair they form was observed for a very large number M of distinct queries (M=12,000 in our experiments). The resulting database consists of 500 pairs of interchangeable terms, some examples of which are shown in Table 8. Note that we do not claim here to generate any general purpose kind of thesaurus, we simply identify pairs of terms that seem not to bring any discriminatory value in generating questions from our closed set of templates.

A deeper examination of this set of interchangeable terms reveals several types of pairs of interchangeable terms. First, we see some of the terms that were involved in the questions that were identified as redundant in our preliminary manual analysis, *e.g. can*/*could*, *could*/*should* and *a*/*the*. Few of the pairs are typical synonyms, *e.g. type*/*kind*, and abbreviations, *e.g. u*/*you* and *ok*/*okay*. Other pairs correspond to delete operations and involve terms that can be removed without changing the meaning of the question, *e.g. really*/- and *exactly*/-. Most of the pairs are formed by terms that are related, yet are far from being considered synonyms. These pairs include: (a) adjectives and adverbs that indicate different qualifier strengths, and even opposite sentiments, *e.g. cool*/*cute*, *okay*/*wrong* and *bad*/*weird*; (b) nouns and verbs, which under a certain question context, address the same need, *e.g. do*/*don't*, *get*/*download*, *girls*/*guys* and *definition*/*meaning*; and (c) interchangeable deter-

Filter	Top-3	Top-5
first-word	1.20	2.51
edit-distance	1.02	2.13

Table 9: Average number of deleted questions among top suggestions, for each filter

miners, prepositions and pronouns, *e.g. what/which*, *if/when*, or *a/the*. Finally, we noticed some incorrect pairs, inferred from templates derived from very domain-specific questions and whose application in a more general context might significantly change the intent of the asker *e.g.*, *legal/possible*, *bad/much*. Yet overall using such a list helps with diversification as demonstrated in the next section.

6. DIVERSIFICATION EVALUATION

We first present a qualitative comparison between the *first-word* and *edit-distance* filters introduced in the previous section. We then review the results of the manual and automatic experiments we conducted, by applying these filters on the candidate questions generated by the reranking component. As in our previous evaluations, we focus our analysis on the top suggested questions, namely the short list that should eventually be shown to users.

6.1 Filters Comparison

We applied the *first-word* and *edit-distance* filters to a set of 140,000 queries from our development set and measured how many questions were filtered out. Table 9 presents the average number of deletions in the original top-3 and top-5 questions.

Obviously, the top question is never deleted, yet deletions occurs immediately afterwards. As shown in the table, both filters delete on average at least one of the two questions in second and third rank. Similarly, at least 2 questions are deleted on average out of the four questions in second to fifth rank. In other words, at least 50% of the top questions are considered redundant by both filters. The first-word filter is more aggressive, as expected, as it considers only on a single word. For example, it may reject diverse questions such as "*is drinking hot coffee* **popular***?*" after seeing "*is drinking hot coffee* **healthy***?*", or "*where can I* **fix** *a wii game console?*" after seeing "*where can I* **buy** *a wii game console*", in which a different intent is captured by a single word (manually highlighted in bold here.) On the other hand, the edit-distance filter depends on the quality of its database of interchangeable pairs. For example, the two questions "*what are the {main/major} characteristics of a pig?*" were not considered redundant because *main/major* were not identified as interchangeable terms.

We also measured how far down into the ranked question list each of the filters needs to progress in order to find 5 questions that are considered different. The first-word filter had to consider on average 16.8 questions, while the edit-distance filter needed on average only 10.9 questions. This distinction is important, since the lower the rank gets, the fewer the chances are that the question is both relevant and grammatically correct. This can be observed from the difference in scores between the first and third ranked questions in Table 4. Thus a higher average number of examined questions will probably have a negative impact on the quality of the selected questions.

6.2 Manual Evaluation

We conducted a small-scale manual evaluation on 50 queries. For each query, we inspected the top 3 questions generated by the reranking algorithm, and compared them to the 3 top questions obtained after applying each filter. We assessed whether each question is of good quality, *i.e.* whether it is both relevant to the query and grammatically correct, as per Section 4.2. We also marked each question as "diverse" if it was not redundant with a higher ranked question.

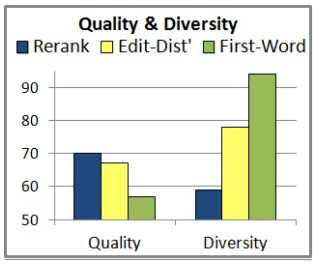

Figure 2: Percentage of good quality questions and of diverse questions before (rerank) and after applying each filter

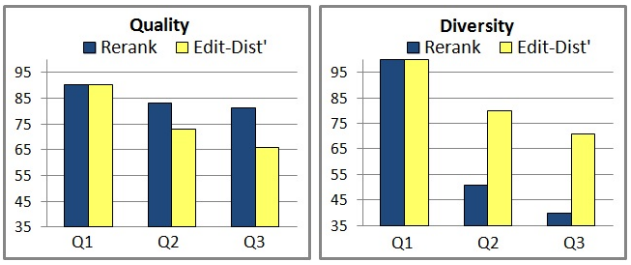

Figure 3: Percentage of good quality questions and of diverse questions at different ranks before (rerank) and after applying each filter

Fig. 2 presents the percentage of good quality questions as well as of diverse questions for a set of 150 evaluated questions (3 questions x 50 queries). The results confirm that the first-word filter, while increasing diversity by 59% over the basic reranking (a growth from 59% to 94%), incurs a significant quality drop of 19% (from 70% quality to 57%). On the other hand, the more conservative edit-distance filter still substantially improves diversity by 32% (from 59% to 78%) with hardly any quality loss (from 70% to 67%).

We extended the manual evaluation to a larger set of 500 queries, assessing only the edit-distance filter. The results, summarized in Fig. 3 show that in the original list, only 45.3% of the questions in ranks #2 and #3 were marked as diverse, but 81.8% of them were found to be of good quality. Filtering for diversity incurred a decrease in quality of 15% (from 82% to 70%). Yet, we believe this is an acceptable loss given the large 66% increase in diversity for the second and third questions (from 45% to 75%).

6.3 Automatic Evaluation

Similarly to our evaluation in Section 4, we wanted a complementary perspective, as provided by a large scale experiment. To this end, we tested both diversification filters under the same automatic experimental setup presented in Section 4.3, in which the algorithms need to rank as high as possible a known (gold-standard) clicked question for each given query. As before, we used as metrics, Recall@1 and Recall@3, which correspond to the percentage of queries for which the gold standard target question is respectively ranked among the top-1 and top-3 questions.

The left plot of Fig. 4 shows the results obtained by each of the diversification filters, as well as the results of the baseline and the reranking components. As mentioned above, the filters have no effect on the first question, therefore the Recall@1 scores are exactly the same for rerank and both filters. The main Recall@3 results show that, while edit-distance performs a little better than first-word, both diversification filters fall behind the reranking algorithm, and their performance is similar to that of the baseline. To better understand this puzzling behavior, we manually inspected some of these suggestions for several queries. We found that in many cases when the top suggestion is not exactly the target question, it is still very similar to it, and conveys the same question intent, *e.g.* "*where can I sell used bikes*" vs. "*where can you sell used bikes*". On several occasions, the reranking algorithm ranks these two variations at the top, since they are similar and receive similar scores. Hence, even if the target question is not the highest ranked question, it is likely to be found in the top 3 suggestions. However, the diversification filters are designed to keep only one of these variations. If they decided to keep the "wrong" variation (not the one marked as target), they will be penalized in our evaluation, even if, in practice, a user would not be bothered by small, intent preserving variations.

Following this observation, we conducted a revised, more lenient, automatic experiment that avoids this pitfall. In this setup, a suggested question is considered as matching the target question if it is "similar enough" to the target question (as opposed to perfectly identical in the previous setup). To this end, we use the edit-distance similarity measure introduced in Section 5.2. As before, we evaluated both diversification filters as well as the reranking and baseline algorithms.

The results of this revised setup are summarized in the right hand side plot of Fig. 4. First, the performance of both the baseline and the reranking algorithms substantially improved as compared to the perfect match setup. Indeed, we see, for Recall@1, an increase of 48% for the baseline and of 37% for the reranking algorithm. For Recall@3, we observe an increase of 26% for the baseline and 16% for the reranking algorithm. Still, the reranking algorithm maintains its superior recall as compared to the baseline, which suggests the improvements are not only due to better matching of small variations. Furthermore, the diversification filters present an impressive boost in performance of 38% for Recall@3 compared to the exact match evaluation. Under this more lenient setup, diversifying the suggestion list via the edit-distance filters shows better results than the reranking algorithm with an increase of 4%. These results indicate that both the baseline and reranking algorithms promote variations of the same question at the top of the suggestion list. While usually the suggestion matches the need of the user (Recall@3 of 85% for the reranking algorithm), adding diversity manages to expose other questions that are sometimes closer to the original intent.

We thus confirm that introducing diversification in the suggestions, using a filter like our edit-distance filter, is beneficial. It incurs only a small decreases in relevance in the strict manual and automatic evaluations, but increases relevance in a more lenient experimental setup.

7. RELATED WORK

The idea of automatically generating natural language questions from queries was first suggested by Lin [14] for two main use cases: improving ranking of search results in existing CQA sites, and offering richer query expansion. Lin further suggested utilizing the existing large amounts of questions in CQA sites along with query logs in order to build a system for question generation given a query, yet without proposing an algorithm for this task. Additionally, Lin suggested an automatic evaluation approach based on gold-standard mapping of queries to clicked questions, in the spirit of the automatic evaluation method we applied here.

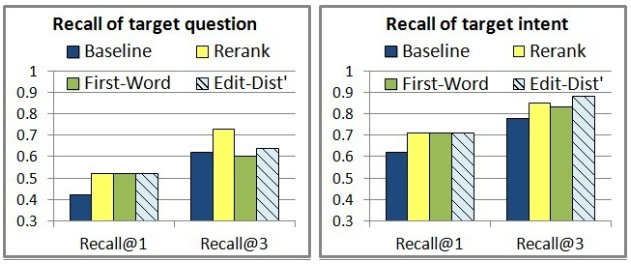

Figure 4: Comparison of the baseline, reranking and diversification methods. Left: Recall of the exact target question. Right: Estimated recall of the target question's intent.

Zhao et al. [24] and Zheng et al. [25] took this idea further. They developed methods for automatically generating questions from queries. Both studies follow the same general template-based approach. Zheng et al. extract templates from existing CQA questions, while Zhao et al. rely on query/question pairs, in which the question is a clicked search result for the query. Then, for a given query, these templates are instantiated and ranked in order to pick the most relevant generated questions for the query. As discussed in the paper, we adopted this approach as well. Our work differs from the above in the candidate ranking method as well as our usage scenario, which imposes different constraints.

Zhao et al. [24] is the closest work to ours, and we indeed used their generation algorithm as a starting point for improvement, as well as our baseline for comparison. In their work, Zhao et al. propose several use cases for automatically generating questions, including improving search in existing CQA archives, automatically posting questions to a CQA site and enabling deeper analysis of query intent. Yet, in none of the proposed scenarios, the questions are intended to be shown to searchers. Our preferred use case precisely consists of serving a choice of suggested synthetic questions, which can then be submitted in one click by the searcher. This scenario imposes additional constraints on the quality of the proposed question, namely higher grammatical correctness, as well as on the diversity of questions, which was completely ignored in [24].

Zheng et al. [25] also use question templates, yet they only extract single-variable templates. This approach relies heavily on existing questions, since new questions can be produced only by replacing a single word in an existing question. It thus imposes a strong limitation on the ability to generate a rich set of questions for new unseen queries, specifically for long tail queries that are the typical candidates for question suggestion. On the other hand, when replacing only a single content word, grammatically correct questions are easier to generate. Therefore, no special effort was made to validate correctness on top of a basic language model. In contrast, our extracted templates count three to five variable slots, which enables far more flexibility in the generation of synthetic questions, but increases the difficulty of achieving correctness. Zheng et al. describe a use case closer to ours, in which a user supplies a few keywords (a query) and is then offered a list of suggested questions. The similarity stops here though, as Zheng et al. envision an interactive dialog, in which at each step the system suggests additional refinement terms that the user may select from. The process is repeated until the user is satisfied with a specific question. While the set of refinement terms is intended to represent a diverse set of user intents, the list of suggested ques-

tions is not explicitly diversified. In contrast, in our use case, the system is allowed to serve a single short question list, and thus diversification considerations have to be incorporated already in this initial (and final) list.

Besides question generation from queries, other question generation tasks were addressed in the literature. The main difference between these tasks lies in the type of textual input from which natural language questions are generated, [22]. These input types include sentences [20, 11, 3], paragraphs [17], and discourse [1, 19], with a variety of applications in mind, such as tutoring, question answering or dialog systems. In contrast to our task of expanding a few query terms into full questions, most of these efforts focus on converting assertions identified in the text into question forms. In addition, due to the differences in goals, none of this efforts considered diversity in a proposed question list, which is one of our key contributions.

8. CONCLUSION

As of today, there is no easy way for searchers, whose needs are not satisfied by Web search engines, to become askers in Community Question Answering sites. We propose here to facilitate such a transition by suggesting to searchers a short list of automatically generated synthetic questions, some of which may have never been asked before, which aim to represent the question intent of the query. Ideally, one of these questions would then be posted by the user in one click, or after minimal editing, on a CQA site. While a few previous studies have explored the automatic generation of questions from queries, we believe that they did not pay enough attention to the correctness of the generated questions. We argue here that this is a prerequisite, if the questions are eventually shown to users, as in our use case.

We introduce a new model for synthetic question generation that pays special attention to the eventual grammatical correctness and appeal of the suggestion list. We use a learning-to-rank framework that leverages millions of features covering both relevance and correctness aspects, which are induced from various syntactic analyses of the candidate questions. We start by extracting question templates from millions of automatically generated query/question pairs. For each input query, these templates are instantiated with the query terms, and ranked in a two step procedure. Before displaying the ranked list to users though, we increase the overall appeal of the suggested list by introducing a novel diversification component. This component filters out "redundant" questions, namely variations of questions representing the same intent, via the automatic detection and utilization of context-sensitive interchangeable terms.

We conducted several manual evaluations, as well as automated large-scale evaluations, that showed that we improve both the relevance and the grammatical correctness of the generated questions, as compared to prior work. More specifically, we demonstrated that the introduction of diversification can even slightly improve relevance, as it increases recall by offering a choice of possible intents to the user. Encouraged by the positive results of these extensive offline experiments, we intend, as our next step, to develop a few use cases and associated treatments, and then investigate the reactions of users in an extensive online experiment.

9. REFERENCES

[1] M. Agarwal, R. Shah, and P. Mannem. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, pages 1–9, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 19–26, New York, NY, USA, 2006. ACM.

[3] H. Ali, Y. Chali, and S. Hasan. Automation of question generation from sentences. *Boyer & Piwek (2010)*, pages 58–67, 2010.

[4] R. Boim, T. Milo, and S. Novgorodov. Diversification and refinement in collaborative filtering recommender. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 739–744, New York, NY, USA, 2011. ACM.

[5] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 335–336, New York, NY, USA, 1998. ACM.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *JMLR*, 7:551–585, 2006.

[7] K. Crammer, R. McDonald, and F. Pereira. Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*, 2005.

[8] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure trees. In *LREC*, 2006.

[9] M. Drosou and E. Pitoura. Search result diversification. *SIGMOD Rec.*, 39(1):41–47, Sept. 2010.

[10] H. D. III, K. Knight, I. Langkilde-geary, D. Marcu, and K. Yamada. The importance of lexicalized syntax models for natural language generation tasks. In *In Proceedings of the 2002 International Conference on Natural Language Generation (INLG - 2002*, pages 9–16, 2002.

[11] S. Kalady, A. Elikkottil, and R. Das. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10, 2010.

[12] T. Lau and E. Horvitz. Patterns of search: analyzing and modeling web query refinement. In *Proceedings of the seventh international conference on User modeling*, UM '99, pages 119–128, Secaucus, NJ, USA, 1999. Springer-Verlag New York, Inc.

[13] J. Lee and S. Seneff. Automatic grammar correction for second-language learners. In *INTERSPEECH*. ISCA, 2006.

[14] C. Lin. Automatic question generation from queries. In *Workshop on the Question Generation Shared Task*, 2008.

[15] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 415–424, New York, NY, USA, 2011. ACM.

[16] Q. Liu, E. Agichtein, G. Dror, Y. Maarek, and I. Szpektor. When web search fails, searchers become askers: understanding the transition. In *Proceedings of the 35th international ACM SIGIR conference on Research and*

development in information retrieval, SIGIR '12, pages 801–810, New York, NY, USA, 2012. ACM.

[17] P. Mannem, R. Prasad, and A. Joshi. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91, 2010.

[18] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98. Association for Computational Linguistics, 2005.

[19] A. Olney, A. Graesser, and N. Person. Question generation from concept maps. *Dialogue & Discourse*, 3(2):75–99, 2012.

[20] S. Pal, T. Mondal, P. Pakray, D. Das, and S. Bandyopadhyay. Qgstec system description–juqgg: A rule based approach. *Boyer & Piwek (2010)*, pages 76–79, 2010.

[21] A. Pauls and D. Klein. Faster and smaller N-gram language models. In D. Lin, Y. Matsumoto, and R. Mihalcea, editors, *Proceedings of the 49th Annual Meeting of the Association of Computational Linguistics*, pages 258–267. The Association for Computer Linguistics, 2011.

[22] V. Rus, B. Wyse, P. Piwek, M. C. Lintean, S. Stoyanchev, and C. Moldovan. The first question generation shared task evaluation challenge. In J. D. Kelleher, B. M. Namee, I. van der Sluis, A. Belz, A. Gatt, and A. Koller, editors, *INLG 2010 - Proceedings of the Sixth International Natural Language Generation Conference*. The Association for Computer Linguistics, 2010.

[23] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. Recommendation diversification using explanations. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 1299–1302, Washington, DC, USA, 2009. IEEE Computer Society.

[24] S. Zhao, H. Wang, C. Li, T. Liu, and Y. Guan. Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 929–937, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[25] Z. Zheng, X. Si, E. Chang, and X. Zhu. K2q: Generating natural language questions from keywords with user refinements. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 947–955, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[26] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.