Domain Dependent Query Reformulation for Web Search

Van Dang
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
vdang@cs.umass.edu

Giridhar Kumaran, Adam Troy
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052, USA
{giridhar, adtroy}@microsoft.com

ABSTRACT

Query reformulation has been studied as a domain independent task. Existing work attempts to expand a query or substitute its terms with the same set of candidates regardless of the domain of this query. Since terms might be semantically related in one domain but not in others, it is more effective to provide candidates for queries with respect to their domain. This paper demonstrates the advantage of this domain dependent query reformulation approach, which learns its candidates, using a standard technique, for each domain from a separate sample of data derived automatically from a generic query log. Our results show that our approach statistically significantly outperforms the domain independent approach, which learns to reformulate from the same log using the same technique, on a large query set consisting of both *health* and *commerce* queries. Our results have very practical interpretation: while building different reformulation systems to handle queries from different domains does not require additional manual effort, it provides substantially better retrieval effectiveness than having a single system handling all queries. Additionally, we show that leveraging domain specific manually labelled data leads to further improvement.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Query Formulation

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Query reformulation, query log, domain dependent.

1. INTRODUCTION

Query reformulation techniques aim to modify user queries to make them more effective for retrieval. Generally, they first identify candidate terms that are similar to the words in the query and then either expand the query with these candidates or substitute them for the original query terms [8]. Existing work has examined different levels of similarity, which results in different types of reformulation. For example, while spelling correction [1, 5] limits candidates to be those that are likely the correct spelling of the misspelled query word, stemming expands a query with morphological variants of its terms [23]. Broader reformulation relaxes the literal constraints and look for candidates that are semantically related to the query terms [16, 27, 8].

Table 1: Examples of *commerce* queries and their reformulations provided by a commercial search engine. Expanding these queries with generic synonyms hurts NDCG.

Query	NDCG@1
flat screen tv	100
flat screen tv +*television*	42.86
car accessories	46.67
car +*auto* accessories	20

None of the techniques above, however, consider the task as domain dependent. This might prove problematic. Terms that can be used interchangeably in general context might not be good candidates for reformulating queries in some specific domains. Table 1 shows two examples for such terms. In general, 'tv' and 'television' are certainly synonyms of each other. In the domain of *commerce* queries, however, expanding the query from 'flat screen tv' to 'flat screen tv *television*' drastically hurts *NDCG* since "television" is rarely used in tv-related commercials. This motivates us to study the domain dependent query reformulation approach that provides different candidate terms for queries in different domains.

In this paper, we investigate the effectiveness of the domain dependent query reformulation approach using a combination of standard reformulation techniques [27, 25]. Instead of training a single reformulation system from a generic query log, we extract <query, clicked document> pairs from the log for each of our target domains, namely *health* and *commerce*, based on the domain of the document. We then train each system for each of the domains from the respective data (Section 3). Assuming the availability of an off-the-shelve query domain classifier, we demonstrate that our domain dependent reformulation approach obtains significantly higher NDCG@1 than the domain independent approach on a large query set containing queries from both domains (Section 4). This improvement, which is consistent across both domains, is due to the fact that our system does not provide as many generic candidates. We also show that leveraging domain specific manually labelled data can provide larger improvement.

We describe our reformulation technique in Section 2. It uses a statistical translation technique to "translate" a query term to its semantically related candidates [25]. These candidates are then filtered by a binary boosted decision tree classifier (Section 2.5). Finally, the resulting candidates are then used by a reformulation model [27] to generate reformulations (Section 2.3 and 2.4). The translation model is estimated from a pseudo parallel corpus derived from a query log (Section 2.1).

The main contributions of this paper are (1) the first large scale study of domain dependent query reformulation for web search (2) we show that the domain dependent reformulation approach is statistically significantly better than the domain independent approach although both systems are trained from the same query log using the same technique. These contributions have important application in search engine practices: while building different reformulation systems to handle queries from different domains costs about the same amount of effort, it is much more effective for retrieval than having a single system handling all incoming queries from the web.

2. METHOD

We view the task of providing a candidate for a query term as "translating" the term to the candidate. Most of the existing work for estimating these translation probabilities in the machine translation literature [4] requires access to parallel corpora, which consists of pairs of sentences: one in the source language and the other in the target language.

In this section, we first describe the process of constructing a *pseudo* parallel corpus from a query log. We then present how to estimate the translation model from this corpus. Finally, we introduce the query reformulation model which modifies user queries based on the candidates suggested by the translation model.

2.1 Pseudo Parallel Corpus Generation

Statistical machine translation (SMT) techniques aim to translate text in the one language (the source language) to another language (the target language). SMT systems often learn from manually created parallel corpora which provide them with a set of sentences in the source language together with their correct translations in the target language [4, 20, 21]. In our context, we can also manually compose reformulations for a set of queries to create a pseudo parallel corpus. This task, however, is very expensive.

On the other hand, any form of rewriting of a query can potential serve as a "translation" of that query. Query logs have proven to contain many reformulations created by the users themselves [26, 15] as well as being a very effective resource for techniques that generate suggestions to the users' queries [3, 28, 2, 19]. Therefore, we build our pseudo parallel corpus from pairs of query and its rewriting, referred to as *parallel pairs* henceforth, extracted from a search log.

2.1.1 Consecutive Query Pairs

Intuitively, when a user issues a query and fails to find relevant documents, they often immediately modify their query to make it more effective. Therefore, consecutive query pairs within a user-session are potentially a good source for reformulations. In fact, existing research [26, 15] reveals that user-sessions in query logs contain approximately 50% reformulations crafted by searchers. Analyzing these pairs for reformulation candidates has been shown to generate effective reformulations [16, 25]. As a result, we consider each consecutive query pair a parallel pair. More advanced techniques such as detecting query chains [24] and search missions [17] can yeild query pairs of higher quality, but we did not do so in this paper.

2.1.2 Query and Clicked Document Title

Even though existing work [14] has pointed out click-through data does not necessarily equal absolute relevance, we argue that clicks on documents in many cases still indicates that the documents are relevant to the queries to some extent. In addition, a title of a document is often the concise summary of its content. As a result, we treat the query and the title of the document clicked for that query a parallel pair.

2.1.3 Query Suggestion via Random Walk

There are a large number of log-based techniques that have been proposed for query suggestion. We use one of the most practical methods which is based on random walk on query click graphs [19, 11].

The click graph is a bipartite graph with queries on one side and documents on the other. An edge connecting a query and a document indicates that we have observed clicks for that document-query pair and its weight is the number of clicks. Starting at a query q_i, a user can "walk" to connected documents, each with some probability. At each document, the user then "walks" to other connected queries and the process repeats. The fact that one can "walk" from q_i to q_j indicates that q_j is relevant to q_i and thus, q_j can be potentially suggested to users who issue q_i. Further details of this model can be found in [11]. We record top-100 suggestions provided for each query by this random walk procedure to create 100 parallel pairs.

Regarding the number of steps of the random walk, longer walks can get further away from the starting point. In the context of the click graph, it is unlikely that queries that are too far from the initial make good suggestions. This intuition is also confirmed by Gao et al. [11] that the most effective transitions are often the first ones. Furthermore, longer transitions can also raise the problem of efficiency. As a result, we limit the number of walks to *two* in our experiments.

2.2 Translation Model

Our translation model, which is based on IBM model 1 [4], aims to provide multiple semantically related candidates as "translations" for any given query term. The translation probability indicates similarity between the term and the candidate. The process of finding these translation candidates as well as estimating the translation probabilities is commonly known as the *word alignment* problem in the machine translation literature [4, 20]. This process can be briefly explained as follows.

Let S denote the source sentence and T denote its correct translation. The word alignment problem looks for the most probable alignment between words in S and words in T. Let $a = \{a_1, a_2, ..., a_m\}$ be an alignment between S and T, where $a_j =< s_{(a_j)}, t_j >$ indicates that the word s_{a_j} in the position a_j in the source sentence is aligned with, or translated to, t_j in the target sentence with probability $Tr(t_j|s_{(a_j)})$. It is clear that the most probable alignment is:

$$\hat{a} = argmax_a \prod_{j=1}^{m} Tr(t_j|s_{(a_j)}) \qquad (1)$$

Equation (1) is straight-forward once we know $Tr(t_j|s_{(a_j)})$. The task of estimating word alignment models, however, involves estimating both $Tr(t_j|s_{(a_j)})$ and the most probable alignment at the same time. This is often done with the Expectation Maximization algorithm [4]. In brief, the training starts by setting all translation probability distribution to uniform. It then identifies the most probable alignments for all parallel pairs using Equation (1). After that, it uses these alignments to update the translation probability distribution. This bootstrapping process proceeds until convergence. Details of this training process can be found in [4]. We use this procedure to estimate our translation model (or the word alignment model) on the pseudo parallel corpus generated as described in Section 2.1.

2.3 Reformulation Model

Given a query $q = w_1, ..., w_{i-1}, w_i, w_{i+1}, ..., w_n$ and the candidate s for the query term w_i from the translation model with the translation probability $Tr(s|w_i)$, the reformulation model determines whether or not to accept this candidate based on both how similar s is to w_i and how fit s is to the context of the query. In particular, the fitness of s is given by:

$$P(w_i \to s|q) = Tr(s|w_i) \times P(w_1...w_{i-1}\_w_{i+1}...w_n|s) \qquad (2)$$

Following Wang and Zhai [27], we assume that context words are independent of one another and the query word only depends on the two context words to its left and right. As a result, the second term in equation (2) is simplified to the following:

$$P_{L_2}(w_{i-2}|s) \times P_{L_1}(w_{i-1}|s) \times P_{R_1}(w_{i+1}|s) \times P_{R_2}(w_{i+2}|s)$$

where $P_{L_i}(w|s)$ and $P_{R_i}(w|s)$ are positional context models which provide the probability that w occur at the i-th position away from s to its left and right in a query respectively. Let G denote the general context which can be either L_i or R_i. The probability $P_G(w|s)$ is a linear interpolation between the probability of seeing w in the context of s and the probability of seeing w in the entire collection used for estimation. It is calculated as follows:

$$P_G(w|s) = \lambda \times \frac{f_w}{\sum_{w_i \in C(s)} f_{w_i}} + (1 - \lambda) \times \frac{f_w}{\sum_{w_i} f_{w_i}} \qquad (3)$$

where $C(s)$ indicates the set of words that occur in the context of s.

It is worth noting that process of estimating the context model $P_G(w|s)$ does not need to distinguish between original queries and "translated" queries. Instead, all queries in the corpus are put together, with proper removal of duplicate queries created by the corpus construction process, to form a single collection of queries. Then the context model is estimated from this collection.

2.4 Candidate Queries Generation

With the translation model and reformulation model in place, we now describe the process of generating candidate reformulations for any given query. For any query $q = w_1, w_2, ..., w_n$, the translation model can provide several candidates for each of the query term. Each of these candidates s_{ij} for the query term w_i is accepted if and only if:

$$\frac{P(w_i \to s_{ij}|q)}{P(w_i \to w_i|q)} \geq 1$$

The idea is that we only accept a candidate only if it fits the query context at least as well as the original term. To tolerate noises in probability estimation, we follow [23] and relax this ratio:

$$\frac{P(w_i \to s_{ij}|q)}{P(w_i \to w_i|q)} \geq \theta$$

With the accepted candidate terms, one can use them to expand the queries or to replace the original query terms. Since it is generally safer to keep the original terms, which is also in line with existing research which states that expansion is more effective than substitution [8], we expand the original queries with these accepted candidates.

2.5 Translation Model Enhanced by Learning

Since our pseudo parallel corpus is constructed automatically, it is prone to contain false positives (pairs of unrelated queries). For example, among consecutive query pairs in a user session, the second query might or might not be the reformulation of the first one. In addition, among the suggested queries generated via random walk, it is unclear how many of them are truly relevant to the original query. Judging by these facts, we believe that the portion of unrelated query pairs in our corpora is relatively high.

Our main goal in this paper is to study what benefit, if there is any, can performing domain dependent query reformulation provide as opposed to having a single system handling queries for all domains. In order to achieve this goal, we favor a baseline reformulation system that works reasonably well so that we can better isolate the effect of domain specific from training errors. A translation model trained on inaccurate data likely provides bad candidate terms, which certainly works against our goal. Therefore, it is important to have high quality output of the translation model.

As a result, we employ machine learning techniques to filter out bad candidates – words that are not semantically related to the original term. In particular, we train a boosted decision tree classifier which classifies if a <*term,candidate*> pair is desirable. The features we use are provided in Table 2. As for the training data, we randomly sampled 26,000 pairs of <*term, candidate*> from session data obtained from a query log. These pairs are then labeled by human judges as *good* and *bad* samples with the criteria being they could be used interchangeably in general context. Finally, among candidates found by the translation model, only those that are labelled as *good* by this classifier are kept.

3. GENERIC VS. DOMAIN SPECIFIC REFORMULATION

In this paper, we assume the availability of a query domain classifier. Therefore, comparing the domain independent query reformulation approach with the domain dependent

Table 2: Features used by the <term , candidate> classifier.

Token-based	1	edit distance between term and candidate
	2	is one of them a prefix of the other
	3	is one of them a suffix of the other
	4	is one of them a stemming variant of the other (using Porter stemmer)
	5	the difference in length (the number of characters) between term and candidate
Log-based	1	term frequency
	2	candidate frequency
	3	#times the candidate occurs in the last query in a user session
	4	#avg. clicks across queries containing the term
	5	#avg. clicks across queries containing the candidate
	6	avg. click-through rate across queries containing the term
	7	avg. click-through rate across queries containing the candidate
	8	translation probability
	9	term entropy
	10	candidate entropy
	11	#avg. shared urls between two consecutive queries containing term and candidate respectively
	12	#avg. shared hosts
	13	#avg. shared clicked urls
	14	#avg. shared clicked hosts
	15	avg. #times a query (contain the candidate) was clicked but its previous query (contain the term) was not

Table 3: Quality of the reformulations (*NDCG@1*) generated by a commercial search engine for queries in different domains.

Domain	#queries	noalter	alter	Improvement
Music	532	64.92	66.40	+1.48†
Video	3,419	62.09	62.59	+0.50†
Commerce	3,316	61.07	61.48	+0.41
Health	185	63.07	63.28	+0.21

approach is essentially comparing the generic reformulation system with the system trained specifically for each domain, which we will refer to as the *domain-specific* reformulation system. Section 3.1 presents the two domains of interest for evaluation. Section 3.2 describes how to derive the the training data for each domain models from the generic query log, which is used to train the generic model, as well as how to estimate each of the models.

3.1 Domains of Interest

To determine which domains we should investigate, we analyze the performance of a commercial search engine across different domains. In particular, we investigate how the search engine performs with and without reformulating user queries in each of these domains. We use the term `alter` and `noalter` to indicate the former and latter case respectively. We first sample a set of approximately 7,000 queries from a query log. We then judge the documents returned by `noalter` and `alter` for these queries. Finally, we run these queries through a proprietary domain classifier to determine the domain of each query. Table 3 shows *NDCG@1* for both systems on each domain. † indicates that the difference between `alter` and `noalter` is statistically significant at p-value < 0.05. We can see that the improvement of `alter` over `noalter` varies across domains. In particular, its reformulations are very effective for *video* and *music* but less so for *health* and *commerce* queries. Consequently, we chose to evaluate our domain dependent reformulation approach with these two domains.

Table 4: Statistics of the query logs used for candidates mining.

Domain	Web	Health	Commerce
#Total Queries	44.4 B	37.7 M	1.5 B
#Unique Queries	6.6 B	9.3 M	209 M
Avg. Query Length	2.97	3.39	2.45

3.2 Model Estimation

For the generic reformulation system, the parallel corpus is constructed from a 18 month's worth web query log recorded by a commercial search engine. Table 4 shows the statistics for this log. The models are estimated from this corpus as described in Section 2.

The log for estimating each of the domain-specific models is derived from the generic log as follows. We use the most frequently clicked web pages in a vertical corresponding to the domain to compile a list of popular websites for that domain. Then we use the portion of the generic log above with clicks on the sites in each list as the query log for that domain. Table 4 also shows the statistics for the two resulting domain specific query logs.

The domain specific model is then estimated from the derived log using the same reformulation technique. The only technical difference is the generic context model is smoothed with the web model to avoid the sparsity problem:

$$P_G(w|s) = \beta \times P_G^{(DS)}(w|s) + (1-\beta) \times P_G^{(Web)}(w|s)$$

where $P_G^{(DS)}(w|s)$ and $P_G^{(Web)}(w|s)$ are the context models as given by equation (3) estimated from the domain specific log and the generic log respectively.

4. EXPERIMENT

4.1 Experimental Setup

Query Sets. Both the query set for the *health* domain and the one for the *commerce* domain are constructed as follows. We first sample a large quantity of queries based on its frequency from the a generic web query log. We then run them through a proprietary query domain classifier and keep only those that are assigned to the respective domains. Finally, we randomly select 5000 queries from those remaining queries to form the query set for each of the domains. Experiments on each domain are then conducted on the appropriate query set. Finally, we merge these two sets together to form a *web* query set.

Experimental Design. Our baseline retrieval system is a commercial search engine which is configured not to reformulate the user queries, which we will refer to as `noalter`. To begin with, we will evaluate the effectiveness of the boosted decision tree classifier at filtering the output of the translation model to decide if it should be a part of our reformulation framework. We then evaluate our domain dependent reformulation approach by comparing each of the domain systems, namely `health` and `commerce`, to the generic systems (`generic`) as well as the `noalter` baseline on the query set from the appropriate domain. We will also refer to both `health` and `commerce` as the `domain` system when there is no need to distinguish between the two.

Each of the reformulation systems under evaluation is used to generate a single reformulation for a given query. `noalter` is used to retrieve result documents for this reformulation. These documents are then judged manually by a separate group of judges on a five-point scale of relevance, from which *NDCG@1* [13] is recorded. We report the performance of each reformulation system on each query set as the *NDCG@1* averaged across all queries in that set. We use the two-tailed t-test with $p\text{-value} < 0.05$ to indicate statistically significant differences.

It is worth noting that the retrieved documents for some reformulations are unjudged since the judges are unable to provide judgment. Thus, for the same query, we might have the judgment for one reformulation system but not for others. Therefore, in the experiments comparing reformulation systems, we use only the subset of queries for which we have judgment for all of the systems involved. As a result, the reported performance of the same system might be slightly different in different experiments since the query sets used, with respect to the systems being compared, can be slightly different.

Due to this experimental setup, the average NDCG score for the whole domain dependent query reformulation system (`DDRS`) on the *web* query set is naturally the average of that of `health` and `commerce` weighted by the number of queries in each set. This is directly comparable with the performance of `generic` on the entire *web* query set.

It is certainly interesting to compare our approach to domain-specific techniques which leverage deeper domain knowledge such as product catalogue and attributes [22, 12]. We leave this for future work since our emphasis is on the fact that applying existing reformulation techniques in a domain dependent manner can be much more helpful than the way they have been used.

Model Parameters. Since our experiments involve manually judging the results, we cannot afford extensive parameter tuning. Instead, we conduct preliminary experiments in which we test a small set of parameter values and choose the ones that work best to apply to the rest of the experiments. As a result, we choose $\lambda = 0.9$, $\beta = 0.9$ and $\theta = 0.9$.

Table 5: Performance comparison between `TM`, `TM-L` and the baseline `noalter`. † and $*$ indicate significant difference to `noalter` and `TM` respectively. "Affected" shows the number of queries each system reformulates. "Imp." and "Win/Loss" provide the improvement in *NDCG@1* and the win/loss ratio of each system over the baseline `noalter`.

		Affected	NDCG	Imp.	Win/Loss
Health	`noalter`		77.95		
	`TM`	1264	77.88	-0.07	76/83
	`TM-L`	201	**78.27**$^{\dagger *}$	+0.32	66/44
Comm.	`noalter`		56.25		
	`TM`	1981	56.06	-0.19	148/175
	`TM-L`	851	**56.42***	+0.17	108/100

4.2 Experimental Results

4.2.1 *Translation Model Filtering: Effectiveness*

As shown in Section 2, our expansion model can generate reformulations for any query by using either candidates provided by the unfiltered translation model (`TM`) or the translation model enhanced by learning (`TM-L`). In this experiment, we compare these two systems with the `noalter` baseline on the *health* and the *commerce* query sets. It should be noted that both translation models are built from web data. We do not use domain data in this experiment since the purpose is to study how well our reformulation technique works on domain specific queries, and how much the training contributes to this process.

Table 5 shows that on both query sets, the unfiltered translation model although can help a certain number of queries, it always hurts a lot more. As a result, its *NDCG@1* is even lower than `noalter`. This result is, in fact, consistent with existing work [8] which shows that reformulation techniques based on unsupervised candidates mining though can help some queries, it hurts a lot more, resulting in overall loss to not reformulating at all.

The model enhanced with learning, on the other hand, outperforms the other two in both cases. Its improvement over `noalter` is statistically significant on the *health* query set and its improvement over `TM` is significant on both query sets. In addition, `TM-L` is also more efficient than `TM` since it generally affects less queries, reducing the workload of the search engine. Therefore, from this moment on, we will only consider `TM-L` for the all reformulation systems.

4.2.2 *Generic vs. Domain Specific*

In this section, we compare `generic` (the domain independent reformulation system) with `health` and `commerce` (the domain-specific reformulation systems) on their respective query set, from which we can then compare `generic` to `DDRS` (the complete domain dependent reformulation system) on the entire *web* query set. Table 6 presents the NDCG score each system achieves together with the number of queries it affected and the number of expansion terms augmented to the original query. We also show the Win/Loss ratio of each system over the baseline – the number of queries it helps and hurts compared to the baseline.

`DDRS` is statistically significantly better than `generic` and `noalter`. This clearly demonstrates the effectiveness of our approach: by simply deriving the training data for each do-

Table 6: Performance comparison between `generic`, `health`, `commerce` and the baseline `noalter` on the *health* and *commerce* query set. † and ∗ indicate significant difference to `noalter` and `generic` respectively. "Afft." and "#Terms" show the number of queries reformulated and terms added by each system. "Imp." and "W/L" show the improvement in *NDCG* and the Win/Loss ratio over `noalter`.

Query set: *web*					
	Afft.	#Terms	NDCG	Imp.	W/L
`noalter`			66.98		
`generic`	1172	1,460	67.18	+0.29	181/159
`DDRS`	782	837	**67.36**$^{\dagger *}$	+0.38	176/135

Query set: *health*					
	Afft.	#Terms	NDCG	Imp.	W/L
`noalter`			77.82		
`generic`	208	223	78.06†	+0.24	65/46
`health`	151	153	**78.15**†	+0.33	63/38

Query set: *commerce*					
	Afft.	#Terms	NDCG	Imp.	W/L
`noalter`			56.18		
`generic`	964	1,237	56.33	+0.15	116/113
`commerce`	631	684	**56.59***	+0.41	113/97

main from a generic query log, we can construct a domain dependent reformulation system that significantly outperforms a domain independent system that learns from the same log using the same reformulation technique.

Furthermore, in both domains, the `generic` system not only reformulates more queries than the `domain` systems, it also adds more terms to each query. The number of reformulations as reflected by the Win/Loss ratio that result in changes in *NDCG*, however, is roughly the same for the two systems. This indicates that the `domain` system is more efficient since it reduces the workload of the search engine yet is still able to achieve a better Win/Loss ratio on both query sets. Additionally, even though the `generic` system outperforms the baseline by 0.24 and 0.15 point on the *health* and *commerce* query set respectively, the `domain` system still manages to provide additional 0.08 and 0.26 point.

Table 7 provides a more detailed view of the results. It shows the performance of `generic` and `domain` on the subset of queries where they provide different reformulation alternatives. It also shows the number of queries and the performance of `noalter` in each subset for reference. This analysis shows that on this query subset, the reformulations offered by the the `domain` systems are substantially better. The difference is statistically significant on the *commerce* query set ($p < 0.03$) and very close to significant on the *health* query set ($p < 0.052$).

There are two possible explanations for this improvement. The `domain` systems might be able to provide additional domain-specific candidates or it might eliminate ineffective general candidates provided by the `generic` model. To study how much impact each has on the improvement, we further divide the set of queries on which the two systems provide different strategies into two: the *Add* set and the *Reject* set. The former includes queries for which the `domain` systems are able to provide additional terms, and the latter consists of queries in which the set of candidate terms added by `domain` (if there is any) is a subset of those added by `generic`. The difference in performance between the two systems in each set in presented in Table 8. The + sign indicates improvement of the `domain` model over the `generic` model. The result indicates that even though the `domain` systems introduce quite many new terms, they only leads to very small gain in *NDCG*. The major gain, in fact, comes from the fact that they are more conservative: they effectively reject many general terms suggested by `generic` system. Table 9 provides examples for both of these cases.

Table 7: Performance comparison on queries where the `generic` system and the `domain` system provide different reformulation alternatives. ∗ indicate significant difference at p-value < 0.05. "#q" is the number of queries in this subset. "p" is the *p-value* returned by the t-test between `generic` and `domain`.

Query set: *health* (#q=198)			
	`generic`	`health`	`noalter`
NDCG	73.07	**75.38** ($p = 0.052$)	75.16

Query set: *commerce* (#q=714)			
	`generic`	`commerce`	`noalter`
NDCG	54.79	**56.45*** ($p = 0.03$)	55.72

Table 8: *NDCG* gains of the two `domain` systems on the *Add* and *Reject* sets with respect to `noalter`. The *Reject* set consists of queries in which the set of candidate terms added by the `domain` systems (if there is any) is a subset of those added by the *web* system. The *Add* set consists of queries for which the `domain` systems are able to provide additional terms.

Query set: *Health*		
	#queries	ΔNDCG
Reject	128	+3.16
Add	70	+0.76

Query set: *Commerce*		
	#queries	ΔNDCG
Reject	560	+2.01
Add	154	+0.39

Next, we analyse the performance of our reformulation systems from a different perspective: in relation to the performance of the original queries. We group the original queries by their *NDCG*. In particular, we look at three ranges of *NDCG*: $[0, 25]$ for queries with low performance, $(25, 50]$ for those with medium performance and $(50, 100]$ for those with high performance. It is worth noting that, all of the queries in the $(50, 100]$ range have the absolute *NDCG* score of 100. Thus we will mark it with [100] instead. We now study how many queries each system helps and hurts in each range, the result of which is shown in Table 10.

Unsurprisingly, all of the improvement happens in the low and medium ranges. In addition, the Win/Loss ratio of the two systems are very comparable in these ranges. Reformulating queries with perfect *NDCG* is certainly undesirable, which is where the two systems differ: the `domain` system by generally affecting less queries, hurts 19% and 18% less queries on the *health* and *commerce* query set respectively. It is relatively clear that most of the improvement `domain` systems have over `generic` comes from the fact that they are

Table 9: Examples of reformulations provided by the `generic` system and the `domain` systems for various queries.

`noalter`		`generic`		`domain`	
Orig. Query	NDCG	Reformulation	NDCG	Reformulation	NDCG
gall bladder disease	100	gall bladder #syn(disease problems)	42.86	gall bladder disease	100
h1n1 vaccine	46.67	h1n1 vaccine	46.67	#syn(h1n1 flu) vaccine	100
low sodium foods	100	low sodium #syn(foods diet)	46.67	low #syn(sodium potassium) foods	100
dunham hiking shoes	100	dunham hiking #syn(shoes boots)	0	dunham hiking shoes	100
plus size skirt sets	0	plus size skirt sets	0	plus size #syn(skirt skirts) sets	100
plus size camis	100	plus size #syn(camis cami)	46.67	plus size #syn(camis camisoles)	100

Table 10: Performance comparison between the `generic` system and the `domain` systems on queries in different *NDCG* ranges. Generally, most of the improvement of both `generic` and `domain` over `noalter` occurs in the low and medium *NDCG* ranges. On the high range, `domain` hurts less queries compared to `generic`, resulting in overall superiority.

	NDCG Ranges	#queries	`generic`			`health`		
			#Afft. Queries	Win	Loss	#Afft. Queries	Win	Loss
Health	$[0, 25]$	281	25	22	0	24	22	0
	$(25, 50]$	1397	73	43	9	54	41	8
	$[100]$	2805	110	0	37	73	0	30
	NDCG Ranges	#queries	`generic`			`commerce`		
			#Afft. Queries	Win	Loss	#Afft. Queries	Win	Loss
Commerce	$[0, 25]$	1519	313	86	6	212	85	4
	$(25, 50]$	1083	243	30	30	157	28	30
	$[100]$	1898	408	0	77	265	0	63

very effectively conservative: they are as good as the generic systems when there is room for improvement and they hurt significantly less queries that already perform well on their own.

4.2.3 Failure Analysis and Discussion

In this section, we look at queries for which the `domain` systems have lower *NDCG* than the `generic` system. We factor this performance lost into two cases: (1) the `domain` systems miss candidates provided by the `generic` system, and (2) the `domain` systems provide candidates that are not provided by the `generic` system.

Firstly, we examine the subset of queries for which the `generic` system can provide candidates that the `domain` system cannot. Table 11 shows example of such queries. It is quite obvious that the `domain` model misses several good candidates. Our further investigation suggests that the `domain`'s translation model, in fact, does provide these candidates. It is the reformulation model that rejects them, resulting in undesirable result.

Secondly, we examine the queries for which the `domain` systems provide candidates that the `generic` system does not. It turns out that most of these candidates are for terms that are part of proper names. Table 11 show some examples of these. For instance, "whole foods market" is the name of a grocery store. Changing its name to "whole food market" certainly has negative effect. From a translation model perspective, "food" is a good candidate for "foods". Therefore, this error should also be attributed to the reformulation model.

There are other instances, however, that are less obvious. Table 12 provides some examples for these. Most of these cases involve morphological variants. For example, if changing "buy silver coins" to "buy silver coin" is bad because "coins" should be in plural form, changing "john deere mower" to "john deere mowers" should be good too since "mower' should also be in plural forms for the same reason. However, the latter is not the case. In addition, *signs* $\rightarrow$ *symptoms* and *handgunds* $\rightarrow$ *pistols* are seemingly good translations given the context of the query, yet they hurt *NDCG*. Unfortunately, we could not identify any probable explanation for these cases.

4.2.4 Potential Improvement

We have demonstrated in Section 4.2.1 that, by leveraging machine learning techniques into the candidates mining process, we significantly improve the quality of the candidates. However, we only use the general term candidate pairs as training data. In this experiment, we study whether using of domain specific term candidates as training data results in additional performance gain. Due to limited resources, however, we can only obtain these pairs for the *health* domain. Therefore, we can only conduct our experiments in the *health* domain.

We re-evaluate the `health` system as described in Section 4.2.2, now with its translation model enhanced with the additional training data in the *health* domain. The result with this system, however, is very close to that of the previous `health` system (as presented in Table 6), thus we do not present it here. The reason is, even though the new translation model can provide new candidates, most of them are rejected by the reformulation model.

This reformulation technique, as described in Section 2.3, is indeed a very simple one which is based on unigram context models. The criteria for the model to accept to candidate terms is also very naive. Therefore, it is accounted for most of the errors made by the `domain` systems as shown in Section 4.3.2. A natural question arises: can we expect to

Table 11: Error analysis: Example queries where the `generic` system outperforms the `domain` systems. There are two types of errors introduced by the `domain` systems: (1) the `domain` systems miss good candidates provided by the `generic` system, and (2) the `domain` systems introduce bad candidates that are not provided by the `generic` system. Most of the type (2) errors are due to the fact that the `domain` systems attempt to change proper names.

	`generic`	NDCG	`domain`	NDCG
(1)	#syn(cannon canon) rebel camera	100	cannon rebel camera	46.67
	#syn(dog pet) barriers	46.67	dog barriers	20
	j hunt #syn(lamps lighting)	48.39	j hunt lamps	9.68
	kenmore gas #syn(ranges range stoves)	100	kenmore gas #syn(ranges range)	0
(2)	thomas the train luggage	100	thomas the train #syn(luggage suitcase)	20
	wb cut tobacco	100	#syn(wb web) cut tobacco	0
	george #syn(srait strait) love #syn(songs lyrics)	100	george #syn(srait straight) love songs	42.86
	whole foods market locations	100	whole #syn(foods food) market locations	48.39

Table 12: Examples of queries for which the reformulations provided by the `domain` systems seem at least as effective as the original query yet they hurt *NDCG*.

`Original Query`	NDCG	`domain`'s Reformulation	NDCG
hannah montana clothes	46.67	hannah montana #syn(clothes clothing)	20
45 long colt revolvers	46.67	45 long colt #syn(revolvers revolver)	0
buy silver coins	100	buy silver #syn(coins coin)	0
john deere mower	100	john deere #syn(mower mowers)	9.68
signs of stroke	100	#syn(signs symptoms) of stroke	42.86
taurus usa handguns	100	taurus usa #syn(handguns pistols handgun)	22.58

Table 13: Performance comparison between `generic`, `health`, and the baseline `noalter` on the *health* query set. † and $*$ indicate significant difference to `noalter` and `generic` respectively.

	NDCG	Imp.	Win/Loss
`noalter`	77.96		
`generic`	78.67†	+0.71	187/138
`health`	**78.78**†	+0.82	192/129
`health-DST`	**78.83**†$*$	+0.87	191/125

further improve the `domain` systems with better reformulation models? Unfortunately, we tried higher order contextual models yet we did not observe any significant changes in improvement. Thus we seek to employ a proprietary reformulation model that is deployed in a commercial search engine. This model is based on higher-order contextual n-gram models. It also considers the relationship among candidates for the given query whereas our naive model only considers the relationship between the candidates and the original query terms. Finally, this model is trained with a boosted decision tree with various features in order determine whether to accept a candidate. In the following experiment, we re-evaluate the `domain` system and the `generic` system with this proprietary model in place of our original reformulation model.

Table 13 shows the performance comparison between the baseline `noalter` in which no queries are reformulated, the `generic` system as above and two `domain` systems: one with only general term candidate pairs as training data and another with additional pairs from the *health* domain. Note that the `generic` system and both `domain` systems now use the proprietary reformulation model instead of our model.

Firstly, all systems are statistically significantly better than the baseline. Secondly, the gap between the baseline and `generic` is now increased to 0.71 compared 0.24 as shown in Table 6. The gap between the baseline and the `health` systems is also increased substantially. These show the benefit of having a more advanced reformulation model. Last but not least, the difference between the `health` system with domain-specific training data, marked as `health-DST`, and the `generic` system is now also statistically significant at *p-value* < 0.05. This clearly demonstrates the potential of using domain specific training data in addition to the general training data.

5. RELATED WORK

Even though many query reformulation techniques have been proposed [7, 16, 27, 25, 10], all of them work in a domain independent manner. The focus of this paper, on the other hand, is to demonstrate that applying these techniques in a domain dependent fashion significantly improves retrieval effectiveness. More importantly, we show that such domain dependent reformulation systems can be constructed simply by effectively utilizing the same generic query log used by the domain independent approach. To our knowledge, this paper is the first to investigate domain dependent reformulation.

Past work on query reformulation focuses on mining query logs recorded by commercial search engines. These techniques tackle the task of reformulation at different levels. Some studies concentrate on spelling correction [1, 5, 18, 6] by restricting their candidate terms to be spelling variants of the query terms. Stemming [23], on the other hand, limits its candidates to be stemming variants of the query terms. Broader reformulation removes those literal constraints and expand the query with semantically related words [7, 16, 27, 25] and concepts [9]. Our work falls into the last category with terms as the expansion unit.

In terms of methodology, the reformulation technique used in this paper is a combination of two existing methods [27, 25]. In particular, it is motivated by that of Riezler and Liu [25] in that it applies statistical machine translation techniques [4] to the task of query rewriting. However, instead of "translating" the whole query, we only "translate" a query term to semantically related candidates for expansion, the acceptance of which is determined by an additional reformulation model [27] down the pipeline. In addition, the outputs of the translation model are filtered by a trained boosted decision tree classifier to ensure high quality of the expansion candidates, which leads to effective reformulations.

Furthermore, Riezler and Liu [25] use queries and the returned document snippets as parallel sentences. Snippets are often much longer than the query, which might hamper the translation model learning algorithm. Instead, we construct our parallel corpus from multiple data sources. These sources are (1) consecutive query pairs in user sessions (2) queries and the title of the clicked documents and (3) query and their suggestions. Our first data source is based on the observations that sessions contain 50% of reformulations created by users themselves [26, 15]. The other two are based on our intuition that clicked documents' titles and suggestions for a query can be considered its reformulations. As a result, our parallel corpus consists of query pairs of more comparable length.

It is worth noting that we investigate the task of query reformulation, which is very different from query suggestion. While query suggestion techniques [3, 2, 19, 11] operate at the query level by finding similar queries from the past, query reformulation methods operate at the term level by modifying individual words or phrases of a query. Consequently, suggested queries are often shown to users since they look more natural whereas reformulations are used to do retrieval without the users' knowledge.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we demonstrate the advantage of domain dependent query reformulation over the domain independent approach. More importantly, we show that this advantage can be obtained simply with an effective utilization of a single generic query log which is used to train the domain independent system. The reformulation technique we use first learns a machine translation model that can "translate" a word to semantically related terms from a pseudo parallel corpus derived from query log data. For each query, the system examines all of the "translations" it has and expand the query with those that are close enough to the original terms and well fit into the context of the query. Our main results suggest that using the same reformulation technique, both reformulation systems for the *health* and *commerce* domains, which learn their candidates from a sample of data extracted automatically from a generic log, outperform the generic system that learns from the same log. In addition, the performance of the domain specific system can be further improved by leveraging domain specific training data.

In our experiments, we consider the general problem of reformulation but not any type in particular. It is interesting to examine which types of reformulation benefits the most from domain specific treatment. In fact, we observe that the domain specific translation model provides new <*misspelled, correctly spelled*> term pairs that are not covered by the general model. However, the query sets used in our experiments do not contain these terms, thus its effect on retrieval performance is unclear. In the future, we will look more specifically into the task of spelling correction.

In this paper, we investigate domain specific reformulation in its simplest form: the only difference difference between the web system and the segment system is the data from which their model is estimated. Further more, the domain data is, in fact, a subset of the general web data. We believe exploiting domain specific data sets and incorporating additional domain specific features can provide further improvement. For example, the health section of Wikipedia provides several alternate names of diseases and medical substances. Simple co-occurrence statistics from this domain specific text also seems to be very useful.

7. REFERENCES

[1] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of HLT*, pages 955-962, 2005.

[2] R. Baeza-Yates, C. Hurtado and M. Mendoza. Query Recommendation Using Query Logs in Search Engines. In *The ClustWeb Workshop*, pages 588-596, 2004.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of KDD*, pages 407-416, 2000.

[4] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Computational Linguistics*, 19(2):263-311, 1993.

[5] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, pages 293-300, 2004.

[6] Q. Chen, M. Li, and M. Zhou. Improving query spelling correction using web search results. In *Proceedings of EMNLP-CoNLL*, pages 181-189, 2007.

[7] H. Cui, J.R. Wen, J.Y. Nie, and W.Y. Ma. Probabilistic query expansion using query logs. In *Proceedings* of WWW, pages 325-332, 2002.

[8] V. Dang and W.B. Croft. Query Reformulation Using Anchor Text. In *Proceedings of WSDM*, pages 41-50, 2010.

[9] B.M. Fonseca, G. Paulo, P. Bruno, R.N. Berthier, and Z. Nivio. Concept-based interactive query expansion. In *Proceedings* CIKM, pages 696-703, 2005.

[10] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proceedings SIGIR*, pages 379-386, 2008.

[11] J. Gao, W. Yuan, X. Li, K. Deng and J.Y. Nie. Smoothing Clickthrough Data for Web Search Ranking. In *Proceedings of SIGIR*, pages 355-362, 2009.

[12] S. Gollapudi, S. Ieong, A. Ntoulas and S. Paparizos. Efficient query rewrite for structured web queries. In *Proceedings of CIKM*, pages 2417-2420, 2011.

[13] K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. In *ACM Transactions on Information Systems*, 20(4):422-446, 2002.

[14] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, pages 133-142, 2002.

[15] R. Jones and D.C. Fain. Query Word Deletion Prediction. In *Proceedings of SIGIR*, pages 435-436, 2003.

[16] R. Jones, B. Rey and O. Madani. Generating query Substitutions. In *Proceedings of WWW*, pages 387-396, 2006.

[17] R. Jones, and K.L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of CIKM*, pages 699-708, 2008.

[18] M. Li, M. Zhu, Y. Zhang, and M. Zhou. Exploring distributional similarity based models for query spelling correction. In *Proceedings of COLING-ACL*, pages 1025-1032, 2006.

[19] Q. Mei, D. Zhou and K. Church. Query Suggestion Using Hitting Time. In *Proceedings of CIKM*, pages 469-477, 2008.

[20] F.J. Och and N. Hermann. A systematic comparison of various statistical alignment models. In *Journal* of Computational Linguistics, 29(1):19-51, 2003.

[21] F.J. Och and N. Hermann. The alignment template approach to statistical machine translation. In *Journal* of Computational Linguistics, 30(4):417-449, 2004.

[22] D. Panigrahi and S. Gollapudi. Result enrichment in commerce search using browse trails. In *Proceedings of WSDM*, pages 267-276, 2011.

[23] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *Proceedings of SIGIR*, pages 639-646, 2007.

[24] F. Radlinski and T. Joachims. Query Chains: Learning to Rank from Implicit Feedback. In *Proceedings of KDD*, pages 239-248, 2005.

[25] S. Riezler and Y. Liu. Query Rewriting Using Monolingual Statistical Machine Translation. In *Association for Computational Linguistics*, 36(3):569-582, 2010.

[26] A. Spink, B.J. Jansen, and H. C. Ozmultu. Use of Query Reformulation and Relevance Feedback by Excite Users. In *Internet Research: Electronic Networking Applications and Policy*, 10(4):317-328, 2000.

[27] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of CIKM*, pages 479-488, 2008.

[28] J. Wen, J.Y. Nie and H.J. Zhang. Clustering User Queries of a Search Engine. In *Proceedings of WWW*, pages 162-168, 2001.