Using Search-Logs to Improve Query Tagging

Kuzman Ganchev Keith Hall Ryan McDonald Slav Petrov
Google, Inc.
{kuzman|kbhall|ryanmcd|slav}@google.com

Abstract

Syntactic analysis of search queries is important for a variety of information-retrieval tasks; however, the lack of annotated data makes training query analysis models difficult. We propose a simple, efficient procedure in which part-of-speech tags are transferred from retrieval-result snippets to queries at training time. Unlike previous work, our final model does not require any additional resources at run-time. Compared to a state-of-the-art approach, we achieve more than 20% relative error reduction. Additionally, we annotate a corpus of search queries with part-of-speech tags, providing a resource for future work on syntactic query analysis.

1 Introduction

Syntactic analysis of search queries is important for a variety of tasks including better query refinement, improved matching and better ad targeting (Barr et al., 2008). However, search queries differ substantially from traditional forms of written language (e.g., no capitalization, few function words, fairly free word order, etc.), and are therefore difficult to process with natural language processing tools trained on standard corpora (Barr et al., 2008). In this paper we focus on part-of-speech (POS) tagging queries entered into commercial search engines and compare different strategies for learning from search logs. The search logs consist of user queries and relevant search results retrieved by a search engine. We use a supervised POS tagger to label the result snippets and then transfer the tags to the queries, producing a set of noisy labeled queries. These labeled queries are then added to the training data and the tagger is retrained. We evaluate different strategies for selecting which annotation to transfer and find that using the result that was clicked by the user gives comparable performance to using just the top result or to aggregating over the top-k results.

The most closely related previous work is that of Bendersky et al. (2010, 2011). In their work, unigram POS tag priors generated from a large corpus are blended with information from the top-50 results from a search engine at prediction time. Such an approach has the disadvantage that it necessitates access to a search engine at run-time and is computationally very expensive. We re-implement their method and show that our direct transfer approach is more effective, while being simpler to instrument: since we use information from the search engine only during training, we can train a stand-alone POS tagger that can be run without access to additional resources. We also perform an error analysis and find that most of the remaining errors are due to errors in POS tagging of the snippets.

2 Direct Transfer

The main intuition behind our work, Bendersky et al. (2010) and Rüd et al. (2011), is that standard NLP annotation tools work better on snippets returned by a search engine than on user supplied queries. This is because snippets are typically well-formed English sentences, while queries are not. Our goal is to leverage this observation and use a supervised POS tagger trained on regular English sentences to generate annotations for a large set of queries that can be used for training a query-specific model. Perhaps the simplest approach – but also a surprisingly powerful one – is to POS tag some relevant snippets for

a given query, and then to transfer the tags from the snippet tokens to matching query tokens. This "direct" transfer idea is at the core of all our experiments. In this work, we provide a comparison of techniques for selecting snippets associated with the query, as well as an evaluation of methods for aligning the matching words in the query to those in the selected snippets.

Specifically, for each query[1] with a corresponding set of "relevant snippets," we first apply the baseline tagger to the query and all the snippets. We match any query terms in these snippets, and copy over the POS tag to the matching query term. Note that this can produce multiple labelings as the relevant snippet set can be very diverse and varies even for the same query. We choose the most frequent tagging as the canonical one and add it to our training set. We then train a query tagger on all our training data: the original human annotated English sentences and also the automatically generated query training set.

The simplest way to match query tokens to snippet tokens is to allow a query token to match any snippet token. This can be problematic when we have queries that have a token repeated with different parts-of-speech such as in "tie a tie." To make a more precise matching we try a sequence of matching rules: First, exact match of the query n-gram. Then matching the terms in order, so the query "tie$_a$ a tie$_b$" matched to the snippet "to tie$_1$ a neck tie$_2$" would match *tie$_a$:tie$_1$* and *tie$_b$:tie$_2$*. Finally, we match as many query terms as possible. An early observation showed that when a query term occurs in the result URL, e.g., searching for "irs mileage rate" results in the page `irs.gov`, the query term matching the URL domain name is usually a proper noun. Consequently we add this rule.

In the context of search logs, a relevant snippet set can refer to the top k snippets (including the case where $k = 1$) or the snippet(s) associated with results clicked by users that issued the query. In our experiments we found that different strategies for selecting relevant snippets, such as selecting the snippets of the clicked results, using the top-10 results or using only the top result, perform similarly (see Table 1).

[1]We skip navigational queries, e.g, *amazon* or *amazon.com*, since syntactic analysis of such queries is not useful.

Query	budget/NN rent/VB a/DET car/NN	Clicks
Snip 1	...Budget/NNP Rent/NNP A/NNP Car/NNP ...	2
Snip 2	...Go/VB to/TO Budget/NNP to/TO rent/VB a/DET car/NN ...	1
Snip 3	...Rent/VB a/DET car/NN from/IN Budget/NNP ...	1

Figure 1: Example query and snippets as tagged by a baseline tagger as well as associated clicks.

By contrast Bendersky et al. (2010) use a linear interpolation between a prior probability and the snippet tagging. They define $\pi(t|w)$ as the relative frequency of tag t given by the baseline tagger to word w in some corpus and $\psi(t|w, s)$ as the indicator function for word w in the context of snippet s has tag t. They define the tagging of a word as

$$\arg\max_t \; 0.2\pi(t|w) + 0.8 \operatorname*{mean}_{s:w\in s} \psi(t|w, s) \quad (1)$$

We illustrate the difference between the two approaches in Figure 1. The numbered rows of the table correspond to three snippets (with non-query terms elided). The strategy that uses the clicks to select the tagging would count two examples of "Budget/NNP Rent/NNP A/NNP Car/NNP" and one for each of two other taggings. Note that snippet 1 and the query get different taggings primarily due to orthographic variations. It would then add "budget/NNP rent/NNP a/NNP car/NNP" to its training set. The interpolation approach of Bendersky et al. (2010) would tag the query as "budget/NNP rent/VB a/DET car/NN". To see why this is the case, consider the probability for rent/VB vs rent/NNP. For rent/VB we have $0.2 + 0.8 \times \frac{2}{3}$, while for rent/NNP we have $0 + 0.8 \times \frac{1}{3}$ assuming that $\pi(\text{VB}|\text{rent}) = 1$.

3 Experimental Setup

We assume that we have access to labeled English sentences from the PennTreebank (Marcus et al., 1993) and the QuestionBank (Judge et al., 2006), as well as large amounts of unlabeled search queries. Each query is paired with a set of relevant results represented by snippets (sentence fragments containing the search terms), as well as information about the order in which the results were shown to the user and possibly the result the user clicked on. Note that different sets of results are possible for the

same query, because of personalization and ranking changes over time.

3.1 Evaluation Data

We use two data sets for evaluation. The first is the set of 251 queries from Microsoft search logs (MS-251) used in Bendersky et al. (2010, 2011). The queries are annotated with three POS tags representing nouns, verbs and "other" tags (MS-251 NVX). We additionally refine the annotation to cover 14 POS tags comprising the 12 universal tags of Petrov et al. (2012), as well as proper nouns and a special tag for search operator symbols such as "-" (for excluding the subsequent word). We refer to this evaluation set as MS-251 in our experiments. We had two annotators annotate the whole of the MS-251 data set. Before arbitration, the inter-annotator agreement was 90.2%. As a reference, Barr et al. (2008) report 79.3% when annotating queries with 19 POS tags. We then examined all the instances where the annotators disagreed, and corrected the discrepancy. Our annotations are available at `http://code.google.com/p/query-syntax/`.

The second evaluation set consists of 500 so called "long-tail" queries. These are queries that occurred rarely in the search logs, and are typically difficult to tag because they are searching for less-frequent information. They do not contain navigational queries.

3.2 Baseline Model

We use a linear chain tagger trained with the averaged perceptron (Collins, 2002). We use the following features for our tagger: current word, suffixes and prefixes of length 1 to 3; additionally we use word cluster features (Uszkoreit and Brants, 2008) for the current word, and transition features of the cluster of the current and previous word. When training on Sections 1-18 of the Penn Treebank and testing on sections 22-24, our tagger achieves 97.22% accuracy with the Penn Treebank tag set, which is state-of-the-art for this data set. When we evaluate only on the 14 tags used in our experiments, the accuracy increases to 97.88%.

We experimented with 4 baseline taggers (see Table 2). WSJ corresponds to training on only the standard training sections of Wall Street Journal portion of the Penn Treebank. WSJ+QTB adds the QuestionBank as training data. WSJ NOCASE and WSJ+QTB NOCASE use case-insensitive version of the tagger (conceptually lowercasing the text before training and before applying the tagger). As we will see, all our baseline models are better than the baseline reported in Bendersky et al. (2010); our lowercased baseline model significantly outperforms even their best model.

Method	MS-251 NVX	MS-251	long-tail
DIRECT-CLICK	93.43	84.11	78.15
DIRECT-ALL	93.93	84.39	77.73
DIRECT-TOP-1	93.93	84.60	77.60

Table 1: Evaluation of snippet selection strategies.

4 Experiments

First, we compared different strategies for selecting relevant snippets from which to transfer the tags. These systems are: DIRECT-CLICK, which uses snippets clicked on by users; DIRECT-ALL, which uses all the returned snippets seen by the user;[2] and DIRECT-TOP-1, which uses just the snippet in the top result. Table 1 compares these systems on our three evaluation sets. While DIRECT-ALL and DIRECT-TOP-1 perform best on the MS-251 data sets, DIRECT-CLICK has an advantage on the long tail queries. However, these differences are small ($<$0.6%) suggesting that any strategy for selecting relevant snippet sets will return comparable results when aggregated over large amounts of data.

We then compared our method to the baseline models and a re-implementation of Bendersky et al. (2010), which we denote BSC. We use the same matching scheme for both BSC and our system, including the URL matching described in Section 2. The URL matching improves performance by 0.4-3.0% across all models and evaluation settings.

Table 2 summarizes our final results. For comparison, Bendersky et al. (2010) report 91.6% for their final system, which is comparable to our implementation of their system when the baseline tagger is trained on just the WSJ corpus. Our best system achieves a 21.2% relative reduction in error on their annotations. Some other trends become appar-

[2]Usually 10 results, but more if the user viewed the second page of results.

Method	MS-251 NVX	MS-251	long-tail
WSJ	90.54	75.07	53.06
BSC	91.74	77.82	57.65
DIRECT-CLICK	93.36	**85.81**	76.13
WSJ + QTB	90.18	74.86	53.48
BSC	91.74	77.54	57.65
DIRECT-CLICK	93.01	85.03	76.97
WSJ NOCASE	92.87	81.92	74.31
BSC	93.71	84.32	76.63
DIRECT-CLICK	93.50	84.46	77.48
WSJ + QTB NOCASE	93.08	82.70	74.65
BSC	93.57	83.90	77.27
DIRECT-CLICK	93.43	84.11	**78.15**

Table 2: Tagging accuracies for different baseline settings and two transfer methods.DIRECT-CLICK is the approach we propose (see text). Column MS-251 NVX evaluates with tags from Bendersky et al. (2010). Their baseline is 89.3% and they report 91.6% for their method. MS-251 and Long-tail use tags from Section 3.1. We observe snippets for 2/500 long-tail queries and 31/251 MS-251 queries.

ent in Table 2. Firstly, a large part of the benefit of transfer has to do with case information that is available in the snippets but is missing in the query. The uncased tagger is insensitive to this mismatch and achieves significantly better results than the cased taggers. However, transferring information from the snippets provides additional benefits, significantly improving even the uncased baseline taggers. This is consistent with the analysis in Barr et al. (2008). Finally, we see that the direct transfer method from Section 2 significantly outperforms the method described in Bendersky et al. (2010). Table 3 confirms this trend when focusing on proper nouns, which are particularly difficult to identify in queries.

We also manually examined a set of 40 queries with their associated snippets, for which our best DIRECT-CLICK system made mistakes. In 32 cases, the errors in the query tagging could be traced back to errors in the snippet tagging. A better snippet tagger could alleviate that problem. In the remaining 8 cases there were problems with the matching – either the mis-tagged word was not found at all, or it was matched incorrectly. For example one of the results for the query "bell helmet" had a snippet containing "Bell cycling helmets" and we failed to match helmet to helmets.

Method	P	R	F
WSJ + QTB NOCASE	72.12	**79.80**	75.77
BSC	82.87	69.05	75.33
BSC + URL	**83.01**	70.80	76.42
DIRECT-CLICK	79.57	76.51	**78.01**
DIRECT-ALL	75.88	78.38	77.11
DIRECT-TOP-1	78.38	76.40	77.38

Table 3: Precision and recall of the NNP tag on the long-tail data for the best baseline method and the three transfer methods using that baseline.

5 Related Work

Barr et al. (2008) manually annotate a corpus of 2722 queries with 19 POS tags and use it to train and evaluate POS taggers, and also describe the linguistic structures they find. Unfortunately their data is not available so we cannot use it to compare to their results. Rüd et al. (2011) create features based on search engine results, that they use in an NER system applied to queries. They report report significant improvements when incorporating features from the snippets. In particular, they exploit capitalization and query terms matching URL components; both of which we have used in this work. Li et al. (2009) use clicks in a product data base to train a tagger for product queries, but they do not use snippets and do not annotate syntax. Li (2010) and Manshadi and Li (2009) also work on adding tags to queries, but do not use snippets or search logs as a source of information.

6 Conclusions

We described a simple method for training a search-query POS tagger from search-logs by transferring context from relevant snippet sets to query terms. We compared our approach to previous work, achieving an error reduction of 20%. In contrast to the approach proposed by Bendersky et al. (2010), our approach does not require access to the search engine or index when tagging a new query. By explicitly re-training our final model, it has the ability to pool knowledge from several related queries and incorporate the information into the model parameters. An area for future work is to transfer other syntactic information, such as parse structures or supertags using a similar transfer approach.

References

C. Barr, R. Jones, and M. Regelson. 2008. The linguistic structure of English web-search queries. In *Proc. of EMNLP*.

M. Bendersky, W.B. Croft, and D.A. Smith. 2010. Structural annotation of search queries using pseudo-relevance feedback. In *Proc. of CIKM*.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

J. Judge, A. Cahill, and J. van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proc. of ACL*.

X. Li, Y.Y. Wang, and A. Acero. 2009. Extracting structured information from user queries with semi-supervised conditional random fields. In *Proc. of SIGIR*.

X. Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proc. of ACL*.

M. Manshadi and X. Li. 2009. Semantic tagging of web search queries. In *Proc. of ACL*.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, 19.

S. Petrov, D. Das, and R. McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

S. Rüd, M. Ciaramita, J. Müller, and H. Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proc. of ACL*.

J. Uszkoreit and T. Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proc. of ACL*.