Syntactic Parsing of Web Queries with Question Intent

Yuval Pinter[1], Roi Reichart[1,2], and Idan Szpektor[1]

[1]Yahoo Research, Haifa 31905, Israel, {yuvalp, roiri, idan}@yahoo-inc.com
[2]Faculty of Industrial Engineering and Management, Technion, IIT

Abstract

Accurate automatic processing of Web queries is important for high-quality information retrieval from the Web. While the syntactic structure of a large portion of these queries is trivial, the structure of queries with question intent is much richer. In this paper we therefore address the task of statistical syntactic parsing of such queries. We first show that the standard dependency grammar does not account for the full range of syntactic structures manifested by queries with question intent. To alleviate this issue we extend the dependency grammar to account for segments – independent syntactic units within a potentially larger syntactic structure. We then propose two distant supervision approaches for the task. Both algorithms do not require manually parsed queries for training. Instead, they are trained on millions of *(query, page title)* pairs from the *Community Question Answering (CQA)* domain, where the CQA page was clicked by the user who initiated the query in a search engine. Experiments on a new treebank[1] consisting of 5,000 Web queries from the CQA domain, manually parsed using the proposed grammar, show that our algorithms outperform alternative approaches trained on various sources: tens of thousands of manually parsed OntoNotes sentences, millions of unlabeled CQA queries and thousands of manually segmented CQA queries.

1 Introduction

As the World Wide Web grows in volume, it encompasses ever-increasing amounts of text. A major gateway to this invaluable resource is *Web queries* which users compose to guide a search engine in retrieving the information they desire to inspect. Automatic processing of Web queries is therefore of utmost importance.

Previous research (Bergsma and Wang, 2007; Barr et al., 2008) suggested that many Web queries are trivial in structure, usually embodying entity lookup, e.g. "*frozen*" or "*condos in NY*". However, with the increasing popularity of Community Question Answering (CQA) sites, such as Yahoo Answers, StackOverflow and social QA forums, more Web queries encompass information needs related to questions that these sites can answer. We found that this subcategory of queries, which we call *CQA queries* (following Liu et al. (2011; Carmel et al. (2014)), exhibits a wide range of structures. This suggests that the processing of such queries, which constitute $\sim$10% of all queries issued to search engines (White et al., 2015), may benefit from syntactic analysis (Tsur et al., 2016).

Recent progress in statistical parsing (Choi et al., 2015) has resulted in models that are both fast, parsing several hundred sentences per second, and accurate. These parsers, however, still suffer from the problem of domain adaptation (McClosky et al., 2010), excelling mostly when their training and test domains are similar. This problem is of particular importance in the heterogeneous Web (Petrov and McDonald, 2012) and is expected to worsen when addressing queries, due to their non-standard grammatical conventions.

Some recent work addresses the syntactic analysis of User Generated Content (UGC) (Petrov and McDonald, 2012; Eisenstein, 2013; Kong et al., 2014). Yet, these efforts generally focus on UGC aspects related to grammatical mistakes made by users (Foster et al., 2008) and to the unique writing conventions of specific Web platforms, such as Twitter (Foster et al., 2011; Kong et al., 2014). Our analysis of thousands of CQA queries, however, reveals that regardless of such issues, CQA queries are generated by a well-defined grammar that sometimes deviates from the one used to generate the standard written language of edited resources such as newspapers.

Consequently, this work has two main contributions. First, we extend the standard dependency grammar to de-

[1]This treebank has been released via Yahoo's Webscope program: `webscope.sandbox.yahoo.com`.

Proceedings of NAACL-HLT 2016, pages 670–680,
San Diego, California, June 12-17, 2016.

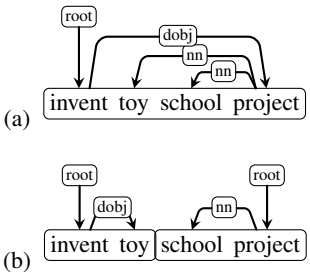

Figure 1: A two-segment query, parsed by an off-the shelf parser: (a) as is, producing an incorrect noun phrase ("toy school project"); (b) after correct segmentation.

scribe the syntax of queries with question intent. The extended grammar is driven by the concept of a *syntactic segment*: an independent syntactic unit within a potentially larger syntactic structure. A query may include several segments, potentially related to each other semantically but lacking an explicit syntactic connection. Hence, query analysis consists of the query's segments and their internal dependency structure, and may be complemented by the inter-segment semantic relationships. Therefore, we constructed a new query treebank consisting of 5,000 CQA queries, manually annotated according to our extended grammar. A comparison of direct application of an off-the-shelf parser (Clear (Choi and McCallum, 2013)) trained on edited text (OntoNotes 5 (Weischedel et al., 2013)) to a raw query with the application of the same parser to the gold-standard segments of that query is given in Fig. 1.

Second, we develop two CQA query parsing algorithms that can adapt any given off-the-shelf dependency parser trained on standard edited text to produce syntactic structures that conform to the extended grammar. Both our algorithms employ *distant supervision* in the form of a training set consisting of millions of *(query, title)* pairs. The *title* is the title of the Yahoo Answers question page that was clicked by the user who initiated the *query*. The alignment between the query and the title provides a valuable training signal for query segmentation and parsing, since the title is usually a grammatical question. Both algorithms employ an off-the-shelf parser, but differ on whether segmentation and parsing are performed in a pipeline or jointly.

We compared our algorithms to several alternatives: (a) Direct application of an off-the-shelf parser to queries; (b) A supervised variant of our pipeline algorithm where thousands of manually segmented queries replace the distant supervision source; and (c) A pipeline algorithm similar to ours where segmentation is based on the predictions of a query language model. We report results on two query treebank tasks: (a) Dependency parsing, reporting Unlabeled Attachment Score (UAS); and (b) Query segmentation, which reflects the core aspect of the extended grammar compared to the standard one.

In experiments on our new treebank, our joint model outperformed the alternatives on UAS for the full test set and for the subset of single-segment queries. Our pipeline model excelled both on UAS and on segmentation F1 for two large subsets that are automatically identifiable at test time: (a) Queries that consist mostly of content words (42.4% of the test set); and (b) Queries for which the confidence score of the off-the-shelf parser is at most 0.8 (30% of the test set). It also beat all other models on the subset of multi-segment queries.

2 Previous Work

The Web attracts considerable NLP research attention (e.g. Eisenstein (2013)). Here we focus on grammar and parsing of Web data in general and queries in particular.

Syntactic Query Analysis Web queries differ from standard sentences in a number of aspects: they tend to be shorter, not to follow standard grammatical conventions, and to convey more information than can be directly inferred from their words. Consequently, a number of works addressed their syntactic analysis. Allan and Raghavan (2002) use part-of-speech (POS) tag patterns in order to manually map very short queries into clarification questions, which are then presented to the user to help them clarify their intent. Barr et al. (2008) trained POS taggers for Web queries and used a set of rules to map the resulting tagged queries into one of seven syntactic categories, whose merit is tested in the context of information retrieval tasks. Manshadi and Li (2009) and Li (2010) addressed the task of semantic tagging and structural analysis of Web queries, focusing on noun phrase queries. Bendersky et al. (2010) used the POS tags of the top-retrieved documents to enhance the initial POS tagging of query terms. Bendersky et al. (2011) proposed a joint framework for annotating queries with POS tags and phrase chunks. Ganchev et al. (2012) trained a POS tagger on automatically tagged queries. The POS tags of the training queries are projected from sentences containing the query terms within Web pages retrieved for them. The retrieved sentences were POS tagged using an off-the-shelf tagger. These works, as opposed to ours, do not aim to produce a complete syntactic analysis of queries.

Syntactic Parsing of Web Data To the best of our knowledge, only a handful of works have aimed at building syntactic parsers for Web data. Petrov and McDonald (2012) conducted a shared task on parsing Web data from the Google Web Treebank, consisting of texts from the email, weblog, CQA, newsgroup, and review domains. The participating systems relied mostly on existing domain adaptation techniques to adapt parsers trained on existing treebanks of edited text to the Web. Foster et al.

(2011) took a similar approach for tweet parsing. Contrary to our approach, these works rely on existing grammatical frameworks, particularly phrase-structure and dependency grammars, and do not aim at adapting them to domains such as Web queries, where standard grammar does not properly describe the language. This may be the reason Web queries were not included in the shared task.

A work that is more related to ours is Kong et al. (2014), who addressed the task of tweet parsing. Like us, they adapt the grammatical annotation scheme to the target linguistic domain and produce a multi-rooted syntactic structure. However, CQA queries and tweets exhibit different syntactic properties: (1) tweets often consist of multiple sentences, while CQA queries are concise in nature and usually correspond to a phrase, a fragment of a sentence, or several of these concatenated; and (2) queries are generated in order to retrieve information from the Web. Tweets, on the other hand, usually aim to convey a short message. These differences lead us to take approaches substantially different from theirs.

3 CQA Query Grammar

3.1 Motivating Analysis

In this section we define the class of *CQA queries* and analyze their properties in comparison with other writing genres. This analysis will establish the motivation for the extension of standard dependency grammar so that it accounts for CQA queries (§3.2).

The data we analyzed consists of queries randomly sampled from the Yahoo Answers log. In cases where a searcher, after issuing a Web query on a search engine, viewed a question page on Yahoo Answers, a popular CQA site, this query is logged. From this log we sampled 100K queries for our analysis, as well as 5,000 additional queries for constructing a query treebank (see §3.3).

According to Barr et al. (2008), who analyzed queries from Yahoo's search engine, 69.8% of general Web queries are composed of a single noun phrase, leaving little room for a meaningful taxonomy. Focusing on CQA queries removes this bias and allows exploration of additional syntactic categories of queries. Especially, we would like to delve into the categories Barr et al. label as *word salad* (e.g. "*mp3s free*") and *other-query* (e.g. "*florida reading conference 2006*"), cited as composing 8.1% and 6.8% of all queries respectively and for which no analysis is offered.

To characterize the domain of CQA queries, we compare its properties to those of other Web domain samples: (a) 100K general Web queries; (b) 120K titles of questions posted on Yahoo Answers; and (c) 100K story bodies from Yahoo News. In our analysis, individual sentences were identified using the OpenNLP sentence splitting tool[2], POS-tagged by the Stanford parser (Klein and Manning, 2003)[3] and syntactically parsed using the Clear parser (Choi and McCallum, 2013)[4].

Table 1 presents four measures of syntactic complexity (four leftmost measure columns). The first column reports the average number of word tokens per parsed item. The second column contains the median and mean dependency tree depths, defined as the number of edges in the longest path from the root node to a leaf in the tree. The third and fourth columns present the fraction of dependency tree root edges that go to words POS-tagged as nouns or as verbs, respectively. We use these last two measures as proxies of the syntactic category of the input text, with noun roots often indicating simple noun phrases and verb roots often indicating more complex syntactic forms that include a verb argument structure[5]. Finally, the rightmost column of the table presents the average parser confidence per item provided by Clear, which we use as a proxy for parsing difficulty.

As the table shows, both CQA and general Web queries are harder to parse (according to parser confidence) compared to news article sentences and CQA question titles. Yet, CQA and general Web queries strongly differ with respect to their syntactic complexity. Indeed, CQA queries have substantially more tokens and deeper trees. Moreover, while 62.9% of the root nodes in general query parse trees govern a noun and 30.3% govern a verb, in CQA queries the respective figures are flipped: 32.2% and 62.7%, respectively.

3.2 Dependency Grammar Extension for Queries

Our analysis above reveals the special status of CQA queries. Like general Web queries, CQA queries are hard to parse. However, while the difficulty of parsing general Web queries may result from their short length and shallow syntactic structure, CQA queries are longer and seem to have a deeper syntactic structure.

Based on our manual inspection of thousands of the queries used in the above analysis, we propose an extension of the standard dependency grammar so that it accounts for CQA queries. Our reasoning is that the grammatical structure of CQA queries is a syntactic forest. The query's tokens are partitioned into one or more contiguous *syntactic segments*, each representing a maximal constituent unit syntactically independent of the other units. The final syntactic representation of the query consists of a set of trees produced according to the Stanford Dependency schema (De Marneffe and Manning, 2008),

[2] `opennlp.apache.org`

[3] Manual inspection revealed that this parser outperforms the Stanford tagger (Toutanova et al., 2003) on these sets.

[4] Version 2.0.1, parsing model 1.2. `www.clearnlp.com`

[5] We verified this hypothesis by manual inspection of the 1,000 development set queries (see §3.3).

Corpus	Average token count	Median (mean) tree depth	$root \rightarrow NN*$ edges (%)	$root \rightarrow VB*$ edges (%)	Average parser confidence
News article sentences	18.4	6 (6.5)	12.0	84.8	0.898
CQA question titles	10.5	4 (4.5)	10.0	86.9	0.991
CQA queries	6.4	4 (3.7)	32.2	62.7	0.809
General queries	4.5	3 (3.0)	62.9	30.3	0.752

Table 1: Syntactic properties of four types of Web domains. The four leftmost properties are proxies of syntactic complexity. The fifth property (parser confidence) is a proxy of parsing difficulty. The "$root \rightarrow NN*$ *edges*" and "$root \rightarrow VB*$ *edges*" columns present the fraction of edges from the root of the parse tree that go to words POS-tagged as nouns and verbs respectively.

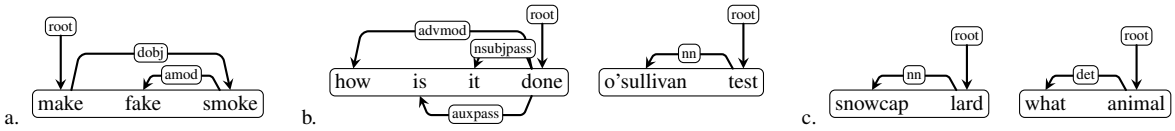

Figure 2: Queries segmented and parsed according to our extended dependency grammar.

one tree per segment. Based on this observation, CQA queries may consist of several syntactically independent segments, as opposed to grammatical sentences which consist of exactly one segment. Importantly, query segments are syntactically independent, although they tend to be semantically related.

Figure 2 provides examples of parsed queries. Query (a) consists of a single segment, a verb phrase rooted by the word *make*. Query (b) is composed of two segments that are syntactically independent, but semantically connected. Particularly, the first segment is an interrogative sentence rooted in the word *done* and the second is a noun phrase which specifies the pronoun *it* from the first segment. Finally, query (c) consists of two segments, each a noun phrase, presumably connected by an *is-made-of* semantic relation. As in query (b), the segments of this query are syntactically independent, but unlike query (b), their semantic connection is more loose. The existence of such loose connections motivates us to exclude semantic subcategorization from the syntactic layer.

We note that the notion of *segment* has another meaning, within the task of *query segmentation* (Bergsma and Wang, 2007; Guo et al., 2008; Tan and Peng, 2008; Mishra et al., 2011; Hagen et al., 2012). This task's goal is to identify words in the query that together form compound concepts or phrases, like "*Chicago Bulls*". As such, this task differs from ours as it defines segmentation in semantic rather than syntactic terms.

We also note that query segments are distinct from the concept of *fragments* in constituency parsing. Marcus et al. (1993) introduce fragments in order to overcome problems involving the attachment point of various modifying phrases, *e.g.* in the sentence '*In Asia, as [FRAG in Europe], a new order is taking shape*'. While proper treatment of such phrases often requires extra syntactic information, their syntactic connection to other parts of the sentence is present, unlike between query segments.

3.3 Query Treebank

Following our proposed grammar, we constructed a treebank by manually annotating 5,000 queries that landed on Yahoo Answers (see §3.1). These queries were randomly split into a 4,000-query test set and a 1,000-query development set. Four human annotators segmented both sets, and parsed the test set with unlabeled dependency trees (including POS tags). The annotators' sets did not overlap, yet cross-reviews were made in cases deemed difficult by them. On a validation set of 100 queries tagged by all annotators, the agreement scores measured were 0.97 for segmentation and 0.96 for dependency edges.

To evaluate how well out-of-the-box parsers conform with our grammar, we applied seven dependency parsers to the 4,000-query test set. We trained all parsers on the OntoNotes 5 training set [6] and applied them: (a) to each gold-standard segment of each query; and (b) to full, unsegmented, queries. For comparison we also recap the performance of the parsers on the OntoNotes 5 test set as reported in Choi et al. (2015). The same non-gold POS tags were provided to all parsers (see §6.2).

The results, presented in Table 2, establish the difficulty of our task: UAS differences between OntoNotes and full queries range from 13.4% to 18.6%. Moreover, injecting gold segmentation knowledge increases the performance of the parsers by 5.3-5.9%, highlighting the value of accurate segmentation in syntactic query analysis. Finally, the ranking of the parsers with respect to their accuracy on queries differs from their ranking with respect to accuracy on OntoNotes, raising Redshift to first place and dropping RBG to last place.

Since knowledge of segment boundaries consistently improves parsing quality in our analysis, we next present two distant supervision approaches that attempt to discover the segments of an input query and return a parse forest that adheres to the segment boundaries.

[6] This parser training convention was kept throughout the paper.

Parser	ON5	Segmented Queries	Full Queries
Redshift (Honnibal et al., 2013)	91.0	**82.5**	**76.6**
Mate (Bohnet and Nivre, 2012)	**91.6**	81.4	75.5
Clear (Choi and McCallum, 2013)	91.3	80.7	75.4
SNN (Chen and Manning, 2014)	88.2	80.7	74.8
GN13 (Goldberg and Nivre, 2013)	89.2	80.5	74.8
Turbo (Martins et al., 2013)	89.6	79.4	73.9
RBG (Lei et al., 2014)	91.4	78.5	72.8

Table 2: UAS of out-of-the-box parsers trained on OntoNotes 5. RBG is reported on its best-performing setting, *basic*.

	Question Title	Associated Query (After segmentation)	Segmentation Cue
1	How many crickets to feed 7 month old leopard gecko?	[leopard gecko] [7 month old] [how many to feed]	Reordering (x2)
2	Does any1 think that Heath Ledger is Cute?	[heath ledger] [cute]	Intruding BE
3	Does Beijing still have license plate restrictions?	[beijing] [license plate restrictions]	Intruding HAVE
4	Do you know the song "Little Sister" by Queens of the Stone Age?	[little sister] [queens]	Intruding IN
5	What came first the jedi or the sith?	[what came first jedi] [sith]	Intruding CC
6	What do you think of a double major in Finance and Marketing?	[double major] [finance] [marketing]	Intruding IN, CC

Table 3: Examples for title-query pairs with the resulting segmentation and the corresponding segmentation cues. Note that they do not necessarily produce correct data.

4 A Pipeline Segmentation-Parsing Model

As our first approach, we present a query segmentation algorithm which can be combined with an off-the-shelf parser in two ways in order to form a pipeline query parsing system: (a) first segmenting the query and then applying the parser to each of the segments; or (b) first parsing the query and then fixing the resulting dependency tree so that it conforms with the segmentation. In the second setup the parser's output is aligned against the segmentation such that dependency edges which cross segment boundaries are re-assigned to become *root* edges. In our experiments we found that, for all our tested pipeline models, the first setup performed slightly better. We therefore report results only for this setup.

Our segmentation algorithm is based on the observation that CQA queries are generated in order to express a searcher's need which is likely to be formulated as a question on the web page that they then visit, and that this paraphrasing can serve as a source for query segmentation cues. The algorithm therefore inspects at training time *(query,title)* pairs where *title* is the title of a Yahoo Answers question page that was clicked by the user who initiated *query*. A query for which a full word-wise alignment to the clicked question can be found is annotated with segmentation markers according to several cues, then added to the training set.

Segmentation Cues The following cues are used for detection of query segment boundaries:

- **Reordering**: a part of the query which appears in the title out of order relative to another part of the query is marked as a segmentation location on both ends (ex. #1 in Table 3). This rule accounted for about 32% of the segmentation cues.
- **Intruding word classes**: if between two words which appear adjacent in the query there are words of certain classes in the question title, the position between them is marked for segmentation. These classes include the verb BE and its conjugations (ex. #2), HAVE and its conjugations (ex. #3), prepositions (ex. #4, #6), and conjunctions (ex. #5, #6). In addition, any multi-word intruding sequence that contains a word of these classes in one of the last three positions is construed as a cue.

Training Set Generation We started with a query log for 60 million Yahoo Answers pages. We filtered out titles of more than 20 words, titles that do not start with a question word, and titles that do not have an associated query which after lowercasing and punctuation removal contains only words from the title (as well as possibly '*site:*' terms or the word '*to*'). The remaining 7.5 million queries were automatically segmented according to the above cues. A sample of 100 queries was found to have a segmentation F1 score (defined in §6.2) of 64.5.

Model and Training We trained a linear chain CRF with pairwise potentials (Lafferty et al., 2001)[7] on the set of 7.5M automatically segmented queries. The model employs the following standard features: (a) unigram and bigram word features (± 2 and ± 1 windows around the represented word respectively); (b) unigram and bigram POS features (in a ± 2 window); (c) unigram word+POS features (± 1 window); and (d) distance of word from start and end of query, as well as each distance combined with word and/or POS.

[7] We used CRF++ (`crfpp.googlecode.com`)

Algorithm 1 Projection-based query parsing

```
1: function FINDPROJECTIONBASEDSEGMENTPARSE-
   TREES(QUERY)
2:     question ← Q2Q(query)
3:     parse ← OffTheShelfParse(question)
4:     for Node n in parse do
5:         if n.text does not match token in query then
6:             CollapseAllEdges(n, parse)
7:             parse.remove(n)
8:     segment-trees ← ∅
9:     for Node r in parse.root.children do
10:        segment-trees.add(Tree(r))
       return segment-trees

1: function COLLAPSEALLEDGES(N, PARSE)
2:     in ← n.incomingEdge
3:     for Edge out in n.outgoingEdges do
4:         label ← GenerateLabel(in.type, out.type)
5:         if n.POS is preposition or conjunction then
6:             parse.addEdge(parse.root, out.target, 'root')
7:         else
8:             parse.addEdge(in.source, out.target, label)
```

5 A Joint Projection-based Model

An alternative distant supervision approach is to employ the *(query,title)* pair set in the training of a model that maps queries to natural language questions. These inferred questions are supposedly easier to parse as they follow standard grammatical conventions. A query parsing algorithm that employs such a mapping component has three steps: first, a grammatical question with a similar information need to that of the query is automatically inferred. Then, the inferred question is parsed by an off-the-shelf dependency parser, trained on a grammatical corpus. Finally, the question parse is projected onto the query, inducing the query's multi-rooted syntactic forest. Algorithm 1 presents pseudo-code for the projection-based query parsing algorithm.

The first step maps a given user query to a synthetic natural-language question. For this step we implemented the Query-to-Question (Q2Q) algorithm of Dror et al. (2013). Q2Q maps queries into valid CQA questions by instantiating templates that were extracted out of *(query,title)* pairs taken from the page view log of a CQA site. We trained the Q2Q algorithm on millions of *(query,title)* pairs taken from the Yahoo Answers log, where each *title* starts with a question word and query length ≥ 3.

Our algorithm obtains the top inferred Q2Q question and parses it using an off-the-shelf parser, trained on grammatical English sentences (lines 2-3 in Algorithm 1). It then projects the question parse tree onto the original query to generate its syntactic structure, in accordance with the extended dependency grammar (§3.2), as follows. First (lines 4-7), the algorithm traverses the question parse, removing all question tokens that do not appear in the query and reassigning the dependents of each removed token to be headed by its parent. In addition, as prepositions and conjunctions inserted by the Q2Q templates are strong signals of segmentation, parse subtrees governed by such nodes are also treated as separate segments. Following this phase, all remaining question tokens appear in the query. The algorithm is then left with a syntactic forest, since the root node may have multiple children, each one defining a query segment. Lines 8-10 extract these segments and their parse trees.

Figure 3 shows the projection process for two queries, the second of which demonstrates how an added preposition ('*on*') invokes segmentation. We emphasize that the only manually annotated data required for the training of our joint model is a treebank of standard edited text (OntoNotes 5), used for training the off-the-shelf parser.

Since we expect the projected query parse to produce trees for contiguous segments, we only accept the top Q2Q result where the query word order is maintained in the question, and use an off-the-shelf parser which produces projective trees. In addition, the projection algorithm may collapse several edges from the question tree into a single edge within a query segment tree, leaving the resulting label unspecified. In this work we evaluate unlabeled parse trees and therefore defer the treatment of this issue to future work (leaving $GenerateLabel()$ in line 4 of Algorithm 1 unspecified).

6 Experiments

6.1 Models

We evaluated the following models (summarized in Table 4) on our new query treebank (§3.3):

Distant Supervision Models Our proposed pipeline (§4) and joint (§5) models. Our pipeline model, denoted by *DistPipe*, performs query segmentation and then parses each segment. Our projection-based joint model, denoted by *Q2QProj*, parses an inferred question for the query and projects the parse tree on the query. Both models use the Clear parser for parsing. Importantly, both our models do not require any type of manually annotated queries (parsed or segmented) for training.

The Q2Q algorithm employed by our joint model returns a question for only 2954 of the 4000 test set queries. We thus reverted to the full-query parsing baseline when it returned no result.

Baselines The first natural baseline is the Clear parser, which is employed by our models. We note that while Clear is not constrained to output a single segment tree[8], in practice it generated a multi-segment structure for $\sim$2% of the development queries, compared to $\sim$25% of

[8]Personal communication with the authors.

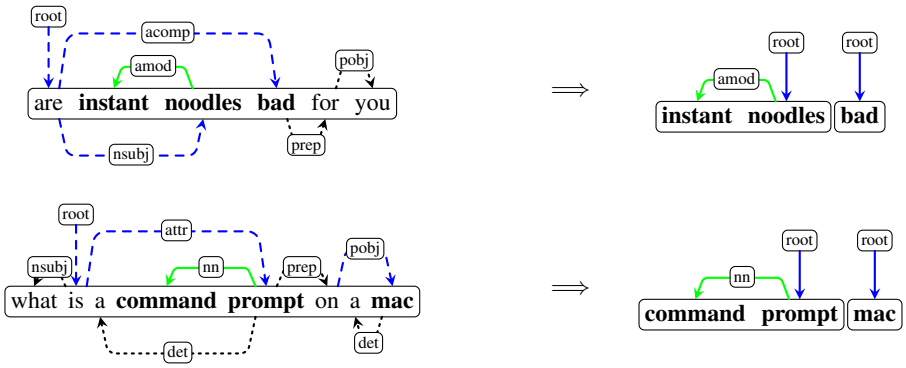

Figure 3: Question parse trees and their projections onto the queries they were generated for. Solid edges are preserved in the projection, dotted edges are removed, and dashed edges are collapsed (in both cases demonstrated, into root edges).

Group	Setup name	Segmentation	Process
Benchmark	*Gold*	gold	pipeline
Ensemble	*Ens*	CRF ensemble	pipeline
Supervised	*Sup*	supervised CRF	pipeline
Distant Supervision	*DistPipe*	distant CRF	pipeline
	Q2QProj	from question parse	projection
Baselines	*Lm*	NNLM	pipeline
	Clear	from parse	parser only

Table 4: The algorithms evaluated in this work.

the queries in the gold standard annotation, making it ill-qualified for multi-segment queries.

As a second baseline, denoted by *Lm*, we constructed a pipeline model identical to ours expect that segmentation is performed with a *Language model (LM)* based approach. For this aim we applied a Neural-Network language model (NNLM) (Mikolov, 2012) to each input query. For each word the language model computes its likelihood given the current model state, which is based on previous words. We then used the 1,000-query development set to find the optimal probability threshold for which words with estimated probability under the threshold are considered "surprising" and therefore mark the beginning of a new segment. We learned a language model[9] with 200 dimensions over 20M randomly sampled queries of length ≥ 3.

Supervised Models We further compare our distant supervision approach to an algorithm that does use manually annotated queries for training, denoted by *Sup*. For this aim we implemented a pipeline model identical to ours, except that the pairwise linear-chain CRF is trained on manually-annotated queries. The algorithm is trained and tested following a 5-fold cross-validation protocol over the 4,000-query test set.

Ensemble Model We also tested the complementary aspects of distant and manual supervision by constructing the same pipeline model, except that segmentation is based on both types of supervision sources. We experimented with various ensemble generation techniques, and the one that has shown to work best was a method that unifies the segmentation decisions of the distant-supervised and supervised CRFs: the model, denoted by *Ens*, considers a token to be a segment boundary if it is considered to be so by at least one of the CRFs.

Gold Segmentation An upper-bound benchmark to our pipeline approach. This is a pipeline model, denoted by *Gold*, that is identical to ours, except that the segmentation is taken from the gold standard.

6.2 Evaluation Tasks and Data Pre-Processing

We consider two evaluation tasks: (a) Dependency parsing, reporting Unlabeled Attachment Score (UAS); and (b) Query Segmentation, reporting the F1 score, where each segment is represented by its boundaries: in order for an observed segment to be considered correct, both of its ends must match those of a gold segment.

All tested algorithms, except for our joint model, directly segment and parse queries and hence require these queries to be POS-tagged. Thus, we POS-tagged the test-set queries with the OpenNLP[10] POS tagger which was adapted to queries using the self-training algorithm of Ganchev et al. (2012). Our self-training set consisted of 14M *(query,title)* pairs from the Yahoo Answers log. This

[9] NNLM implementation in `rnnlm.org` with default parameters.

[10] `opennlp.apache.org`

tagger reached 88.2% accuracy on our query treebank, compared to 81.3% of the off-the-shelf tagger[11].

Analyzing our development set we noticed that a very strong indicator that a query is a grammatical, and thus consists of a single segment, is when it starts with a WH-word or an auxiliary verb. Hence, in all pipeline models, except *Gold*, we do not segment such queries but rather directly apply the Clear parser to them. In postmortem analysis of the test set we found that this indicator was correct for 93.2% of the 1600 detected queries with 40% recall with respect to the single-segment queries subset.

6.3 Results

Tables 5 and 6 present the results of our experiments. The *All Queries* column in Table 5 reports the performance of the tested models on the full test set. Overall, methods based on distant and manual supervision are superior to the baseline methods in both measures. Interestingly, our *Q2QProj* model performs best in terms of UAS (77.1%) although its segmentation F1 is mediocre (63.5). In terms of UAS, the distant-supervised and supervised models approach the performance of the upper bound gold standard segmentation. For example, *Q2QProj* is outperformed by *Gold* by only 3.6%. The *Clear* parser baseline, which is not trained to identify multiple segments, lags 5.7-5.8 F1 points behind the models that employ CRF segmentation. This is translated to a difference in UAS of up to 1.7%. The *Lm* baseline, on the other hand, scores substantially lower than the other models in both measures. This may be an indication of the syntactic, rather than lexical, nature of our task.

In development set experiments we were able to characterize two subsets of queries on which our distant supervised pipeline model performs particularly well, outperforming even the supervised pipeline algorithm which requires thousands of manually annotated queries for training. One such set is the subset of queries that contain at most one word not tagged with what we define to be a *content word POS*, namely: noun, verb, adjective or adverb ('$\leq$ 1 *ncw*' column in Table 5). Intuitively, this subset, which accounts for 42.4% of the test set, consists of queries that convey larger amounts of semantic content and are structured less coherently. On this subset, our *DistPipe* model outperforms the supervised *Sup* model by 4.1 segmentation F1 points, and by 1.7% in UAS.

The other subset, which accounts for 30% of the test set, includes those queries for which the confidence score of the Clear parser, when applied to the whole query, is at most 0.8 ($LowConf$ column in Table 5). This subset singles out cases deemed difficult by the parser, indicating queries with non-standard syntax. Indeed, the performance of all models substantially drops compared to the full set or the '$\leq$ 1 *ncw*' set. Here again, *DistPipe* improves over *Sup* by 5.8 segmentation F1 points and 1.9% in UAS, with additional gain by the ensemble model *Ens*.

Q2QProj performs lower than the pipeline models on both subsets, suggesting that this approach does not work for difficult queries as well as it does for more simple queries. The decreased performance of the *Clear* baseline on both these subsets compared to the entire test set is not surprising, given their challenging syntactic properties.

Altogether, queries belonging to either of the two subsets (or both) account for 50.2% of the full set, emphasizing the benefit of developing more sophisticated ensemble approaches, based on the above characteristics of queries and the individual tested models.

Next, we turn to Table 6, which compares the performance of the various models on the test subsets that consist of single-segment or multi-segment queries only[12]. Our pipeline model, *DistPipe*, excels on the multi-segment subset, achieving a segmentation F1 of 42.2 and UAS of 67.5% compared to only 23.5 and 64.3% respectively of the fully supervised *Sup* model. Our joint model *Q2QProj* achieves a UAS score similar to the supervised model, though its segmentation performance is lower. The ensemble model *Ens* provides additional improvement, hinting that the *Sup* and *DistPipe* models may have learned somewhat different segmentation cues. The segmentation F1 of Clear is as low as 2.2, as its training set contains only single-segment sentences.

We note that the intersection between the multi-segment subset and the '$\leq$ 1 *ncw*' subset is only 622 queries (63.4% of the multi-seg set, 36.7% of '$\leq$ 1 *ncw*'), and with $LowConf$ it is only 524 queries (53.3% of the multi-seg set, 43.7% of $LowConf$). This demonstrates that our *Dist* model is of merit for a variety of query types.

On Single-segment queries, our joint model, *Q2QProj*, achieves the best UAS. This may be because it provides the parser with more context for telegraphic single-segment queries, so much so, that it even outperforms the *Gold* benchmark. Both the *Q2QProj* and *DistPipe* models over-segment (86.5 and 86.8 F1), compared to the near perfect single-segment detection of the supervised *Sup* model (96.2). Still, these differences are only mildly reflected in the UAS scores for single-segment queries.

6.4 Error Analysis

To better understand our distant-supervised signal, we applied the CRF tagger introduced in §4 (without additional filtering) to the test set and analyzed two cases: 100 *false positives* – single-segment queries which were incorrectly tagged with multiple segments, and 100 *false negatives* – multi-segment queries the tagger was wrong to tag as having a single segment.

[11] Several other taggers we experimented with gave similar accuracy figures: ClearNLP, the Stanford tagger and the Stanford parser.

[12] These subsets are extracted using the gold standard and are therefore not available to the models at inference time.

Test Set / Setup	All Queries (N=4000)		≤ 1 *ncw* (N=1694)		*LowConf* (N=1199)	
	F1	UAS	F1	UAS	F1	UAS
Gold	100	80.7	100	81.9	100	73.5
Ens	**70.4**	76.4	**63.5**	**73.0**	**59.2**	**64.6**
Sup	70.3	76	59.1	71.1	52.8	62.4
DistPipe	70.3	76.3	63.2	72.8	58.6	64.3
Q2QProj	63.5	**77.1**	53.1	70.5	47.2	61.1
Lm	37.2	66.6	31.6	59.9	30.8	54.4
Clear	64.6	75.4	51.3	69.5	42.9	60.7

Table 5: Performance of the various models on various automatically-extractable subsets of the data. *All Queries* is our test set, the others are its subsets with: ≤ 1 *ncw* – at most one non-content word according to POS tags; *LowConf* – a Clear parser confidence score of at most 0.8. 'F1' denotes segmentation F1.

Test Set / Setup	Multi-segment (N=984)		Single-segment (N=3016)	
	S. F1	UAS	S. F1	UAS
Gold	100	83.9	100	79.8
Ens	**46.6**	**68.7**	84.8	78.7
Sup	23.5	64.3	96.2	79.4
DistPipe	42.2	67.5	86.8	78.9
Q2QProj	21.2	64.3	86.5	**80.9**
Lm	29.6	60.4	41.6	68.4
Clear	2.2	60.5	**96.9**	79.8

Table 6: Performance on queries which have a single segment or multiple segments according to the gold standard.

Two types of queries cause most false positives cases. The first type, making up 65% of the errors, is a full (or nearly-full) question or sentence. In half of these cases, a word in the middle of the query, which often marks the beginning of a grammatical question, is incorrectly marked as a segment start. Such an example is "*[sherlock is the best show ever]*", where the underlined word is the incorrectly tagged segment start. The other main error is the segmentation of a query consisting of a single noun phrase (19% of the cases), for example "*[clothing product testing]*".

Our false negative analysis discovered four main types of errors, where a multi-segment query is not segmented. The most frequent one (35%) is cases where the tagger did not detect a syntactic cue for a segment start, *e.g.* in "*[grilling pork chops] [seasoning]*", where '*and*' is potentially missing. Another common mistake (24%) is a named entity which is added as its own segment for context (usually in the beginning or the end of the query). Such a query is "*[biotin] [mcg vs mg]*". The third type of errors (17%) is queries for which segmentation detection requires the understanding of the semantics behind the query. One example is "*[movies on youtube] [list]*", where '*youtube list*' was construed as a single noun phrase. The forth type (11%) contains a reference for a preferred content provider by the searcher on its own segment, such as "*[is illuminati good] [yahoo]*".

This analysis shows that the more frequent errors involve semantics and require either a different segmentation approach, or more semantic-oriented features.

7 Conclusions

We studied the syntactic properties of Web queries with question intent. We motivated the need to extend the dependency grammar framework so that it accounts for such queries, and constructed a new Query Treebank, annotated according to the extended grammar. We then developed distant-supervised algorithms that can parse queries according to our grammar. Our algorithms outperform strong baselines, including a supervised model trained on thousands of manually segmented queries.

In future work we would like to improve the query analysis performance of our algorithms. In addition, we plan to assess the contribution of query parsing to IR tasks such as document retrieval and query reformulation.

Acknowledgments

We thank: Bettina Bolla and Shir Givoni, for their annotation work on Query Treebank; Avihai Mejer, for implementing the Q2Q element of the projection algorithm; and Joel Tetreault, for providing the infrastructure for the experiments in Section 3.3.

References

James Allan and Hema Raghavan. 2002. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of SIGIR*.

Cory Barr, Rosie Jones, and Moira Regelson. 2008. The linguistic structure of english web-search queries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Michael Bendersky, W Bruce Croft, and David A Smith. 2010. Structural annotation of search queries using pseudo-

relevance feedback. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1537–1540. ACM.

Michael Bendersky, W Bruce Croft, and David A Smith. 2011. Joint annotation of search queries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *EMNLP-CoNLL*. Citeseer.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP-CoNLL*, pages 1455–1465. Association for Computational Linguistics.

David Carmel, Avihai Mejer, Yuval Pinter, and Idan Szpektor. 2014. Improving term weighting for community question answering search using syntactic analysis. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 351–360.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

Jinho D Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *ACL (1)*, pages 1052–1062.

Jino Choi, Joel Tetreault, and Amanda Stent. 2015. It depends: Dependency parser comparison using a web-based evaluation tool. In *Proceedings of ACL-IJCNLP*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Association for Computational Linguistics.

Gideon Dror, Yoelle Maarek, Avihai Mejer, and Idan Szpektor. 2013. From Query to Question in One Click: Suggesting Synthetic Questions to Searchers. In *Proceedings of WWW 2013*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of NAACL-HLT*.

Jennifer Foster, Joachim Wagner, and Josef van Genabith. 2008. Adapting a wsj-trained parser to grammatically noisy text. In *Proceedings of ACL-HLT: Short Papers*.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, Josef Van Genabith, et al. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *proceedings of the Workshop On Analyzing Microtext (AAAI 2011)*, pages 20–25.

Kuzman Ganchev, Keith Hall, Ryan McDonald, and Slav Petrov. 2012. Using search-logs to improve query tagging. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1:403–414.

Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. 2008. A unified and discriminative model for query refinement. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 379–386. ACM.

Matthias Hagen, Martin Potthast, Anna Beyer, and Benno Stein. 2012. Towards optimum query segmentation: in doubt without. In *Proceedings of CIKM*.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172. Citeseer.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1381–1391.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Qiaoling Liu, Eugene Agichtein, Gideon Dror, Evgeniy Gabrilovich, Yoelle Maarek, Dan Pelleg, and Idan Szpektor. 2011. Predicting web searcher satisfaction with existing community-based answers. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 415–424.

Mehdi Manshadi and Xiao Li. 2009. Semantic tagging of web search queries. In *Proceedings of the ACL-IJCNLP*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*.

André FT Martins, Miguel Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *ACL (2)*, pages 617–622. Citeseer.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Tomáš Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Ph. D. thesis, Brno University of Technology.

Nikita Mishra, Rishiraj Saha Roy, Niloy Ganguly, Srivatsan Laxman, and Monojit Choudhury. 2011. Unsupervised query segmentation using only query logs. In *Proceedings of WWW*, pages 91–92.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59. Citeseer.

Bin Tan and Fuchun Peng. 2008. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*.

Gilad Tsur, Yuval Pinter, Idan Szpektor, and David Carmel. 2016. Identifying web queries with question intent. In *Proceedings of WWW*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston, 2013. *OntoNotes Release 5.0*.

Ryen W. White, Matthew Richardson, and Wen-tau Yih. 2015. Questions vs. queries in informational search tasks. In *Proceedings of WWW'15 Companion*, pages 135–136. WWW.