Enhanced Answer Type Inference from Questions using Sequential Models

Vijay Krishnan and **Sujatha Das** and **Soumen Chakrabarti**[*]
Computer Science and Engineering Department, IIT Bombay, India

Abstract

Question classification is an important step in factual question answering (QA) and other dialog systems. Several attempts have been made to apply statistical machine learning approaches, including Support Vector Machines (SVMs) with sophisticated features and kernels. Curiously, the payoff beyond a simple bag-of-words representation has been small. We show that most questions reveal their class through a short contiguous token subsequence, which we call its *informer span*. Perfect knowledge of informer spans can enhance accuracy from 79.4% to 88% using linear SVMs on standard benchmarks. In contrast, standard heuristics based on shallow pattern-matching give only a 3% improvement, showing that the notion of an informer is non-trivial. Using a novel multi-resolution encoding of the question's parse tree, we induce a Conditional Random Field (CRF) to identify informer spans with about 85% accuracy. Then we build a meta-classifier using a linear SVM on the CRF output, enhancing accuracy to 86.2%, which is better than all published numbers.

1 Introduction

An important step in factual question answering (QA) and other dialog systems is to classify the question (e.g., Who painted Olympia?) to the anticipated type of the answer (e.g., person). This step is called "question classification" or "answer type identification".

The answer type is picked from a hand-built taxonomy having dozens to hundreds of answer types (Harabagiu et al., 2000; Hovy et al., 2001; Kwok et al., 2001; Zheng, 2002; Dumais et al., 2002). QA systems can use the answer type to short-list answer tokens from passages retrieved by an information retrieval (IR) subsystem, or use the type together with other question words to inject IR queries.

Early successful QA systems used manually-constructed sets of rules to map a question to a type, exploiting clues such as the wh-word (who, where, when, how many) and the head of noun phrases associated with the main verb (what is the tallest *mountain* in . . .).

With the increasing popularity of statistical NLP, Li and Roth (2002), Hacioglu and Ward (2003) and Zhang and Lee (2003) used supervised learning for question classification on a data set from UIUC that is now standard[1]. It has 6 coarse and 50 fine answer types in a two-level taxonomy, together with 5500 training and 500 test questions. Webclopedia (Hovy et al., 2001) has also published its taxonomy with over 140 types.

The promise of a machine learning approach is that the QA system builder can now focus on designing features and providing labeled data, rather than coding and maintaining complex heuristic rule-bases. The data sets and learning systems quoted above have made question classification a well-defined and non-trivial subtask of QA for which algorithms can be evaluated precisely, isolating more complex factors at work in a complete QA system.

Prior work: Compared to human performance, the accuracy of question classifiers is not high. In all studies, surprisingly slim gains have resulted from sophisticated design of features and kernels.

Li and Roth (2002) used a Sparse Network of Winnows (SNoW) (Khardon et al., 1999). Their features included tokens, parts of speech (POS), chunks (non-overlapping phrases) and named entity (NE) tags. They achieved 78.8% accuracy for 50 classes, which improved to 84.2% on using an (unpublished, to our knowledge) hand-built dictionary of "semantically related words".

[*] soumen@cse.iitb.ac.in

[1] http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/

Hacioglu and Ward (2003) used linear support vector machines (SVMs) with question word 2-grams and error-correcting output codes (ECOC)—but no NE tagger or related word dictionary—to get 80.2–82% accuracy.

Zhang and Lee (2003) used linear SVMs with all possible question word q-grams, and obtained 79.2% accuracy. They went on to design an ingenious kernel on question parse trees, which yielded visible gains for the 6 coarse labels, but only "slight" gains for the 50 fine classes, because "the syntactic tree does not normally contain the information required to distinguish between the various fine categories within a coarse category".

Algorithm	6-class	50-class
Li and Roth, SNoW	(1)	78.8(2)
Hacioglu et al., SVM+ECOC	–	80.2–82
Zhang & Lee, LinearSVMq	87.4	79.2
Zhang & Lee, TreeSVM	90	–
SVM, "perfect" informer	**94.2**	**88**
SVM, CRF-informer	**93.4**	**86.2**

Table 1: Summary of % accuracy for UIUC data. (1) SNoW accuracy without the related word dictionary was not reported. With the related-word dictionary, it achieved 91%. (2) SNoW with a related-word dictionary achieved 84.2% but the other algorithms did not use it. Our results are summarized in the last two rows, see text for details.

Our contributions: We introduce the notion of the **answer type informer span** of the question (in §2): a short (typically 1–3 word) subsequence of question tokens that are adequate clues for question classification; e.g.: How much does an adult elephant *weigh*?

We show (in §3.2) that a simple linear SVM using features derived from human-annotated informer spans beats all known learning approaches. This confirms our suspicion that the earlier approaches suffered from a feature localization problem.

Of course, informers are useful only if we can find ways to automatically identify informer spans. Surprisingly, syntactic pattern-matching and heuristics widely used in QA systems are not very good at capturing informer spans (§3.3). Therefore, the notion of an informer is non-trivial.

Using a parse of the question sentence, we derive a novel set of multi-resolution features suitable for training a conditional random field (CRF) (Lafferty et al., 2001; Sha and Pereira, 2003). Our feature design paradigm may be of independent interest (§4). Our informer tagger is about 85–87% accurate.

We use a meta-learning framework (Chan and Stolfo, 1993) in which a linear SVM predicts the answer type based on features derived from the original question as well as the output of the CRF. This meta-classifier beats all published numbers on standard question classification benchmarks (§4.4). Table 1 (last two rows) summarizes our main results.

2 Informer overview

Our key insight is that a human can classify a question based on very few tokens gleaned from skeletal syntactic information. This is certainly true of the most trivial classes (*Who* wrote Hamlet? or *How many* dogs pull a sled at Iditarod?) but is also true of more subtle clues (How much does a rhino *weigh*?).

In fact, informal experiments revealed the surprising property that *only one* contiguous span of tokens is adequate for a human to classify a question. E.g., in the above question, a human does not even need the *how much* clue once the word *weigh* is available. In fact, "How much does a rhino *cost*?" has an identical syntax but a completely different answer type, not revealed by *how much* alone. The only exceptions to the single-span hypothesis are multi-function questions like "What is the *name* and *age* of . . .", which should be assigned to multiple answer types. In this paper we consider questions where one type suffices.

Consider another question with multiple clues: *Who* is the *CEO* of IBM? In isolation, the clue *who* merely tells us that the answer might be a person or country or organization, while *CEO* is perfectly precise, rendering *who* unnecessary. All of the above applies *a forteriori* to *what* and *which* clues, which are essentially uninformative on their own, as in "What is the *distance* between Pisa and Rome?"

Conventional QA systems use mild analysis on the wh-clues, and need much more sophistication on the rest of the question (e.g. inferring *author* from *wrote*, and even verb subcategorization). We submit that a single, minimal, suitably-chosen contiguous span of question token/s, defined as the **informer span** of the question, is adequate for question classification.

The informer span is very sensitive to the structure of clauses, phrases and possessives in the question, as is clear from these examples (informers italicized): “What is Bill Clinton’s wife’s *profession*”, and “What *country*’s president was shot at Ford’s Theater”. The choice of informer spans also depends on the target classification system. Initially we wished to handle definition questions separately, and marked no informer tokens in “What is digitalis”. However, *what is* is an excellent informer for the UIUC class `DESC:def` (description, definition).

3 The meta-learning approach

We propose a meta-learning approach (§3.1) in which the SVM can use features from the original question as well as its informer span. We show (§3.2) that human-annotated informer spans lead to large improvements in accuracy. However, we show (§3.3) that simple heuristic extraction rules commonly used in QA systems (e.g. head of noun phrase following wh-word) cannot provide informers that are nearly as useful. This naturally leads us to designing an informer tagger in §4.

Figure 1 shows our meta-learning (Chan and Stolfo, 1993) framework. The combiner is a linear multi-class one-vs-one SVM[2], as in the Zhang and Lee (2003) baseline. We did not use ECOC (Hacioglu and Ward, 2003) because the reported gain is less than 1%.

The word feature extractor selects unigrams and q-grams from the question. In our experience, $q = 1$ or $q = 2$ were best; if unspecified, all possible $q$grams were used. Through tuning, we also found that the SVM “C” parameter (used to trade between training data fit and model complexity) must be set to 300 to achieve their published baseline numbers.

3.1 Adding informer features

We propose two very simple ways to derive features from informers for use with SVMs. Initially, assume that perfect informers are known for all questions; later (§4) we study how to predict informers.

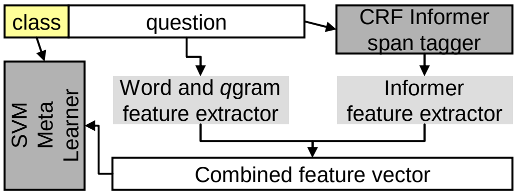

Figure 1: The meta-learning approach.

Informer q-grams: This comprises of all word q-grams within the informer span, for all possible q. E.g., such features enable effective exploitation of informers like *length* or *height* to classify to the `NUMBER:distance` class in the UIUC data.

Informer q-gram hypernyms: For each word or compound within the informer span that is a WordNet noun, we add all hypernyms of all senses. The intuition is that the informer (e.g. *author*, *cricketer*, *CEO*) is often narrower than a broad question class (`HUMAN:individual`). Following hypernym links up to *person* via WordNet produces a more reliably correlated feature.

Given informers, other question words might seem useless to the classifier. However, retaining regular features from other question words is an excellent idea for the following reasons.

First, we kept word sense disambiguation (WSD) outside the scope of this work because WSD entails computation costs, and is unlikely to be reliable on short single-sentence questions. Questions like *How long* . . . or *Which bank* . . . can thus become ambiguous and corrupt the informer hypernym features. Additional question words can often help nail the correct class despite the feature corruption.

Second, while our CRF-based approach to informer span tagging is better than obvious alternatives, it still has a 15% error rate. For the questions where the CRF prediction is wrong, features from non-informer words give the SVM an opportunity to still pick the correct question class.

Word features: Based on the above discussion, one boolean SVM feature is created for every word q-gram over all question tokens. In experiments, we found bigrams ($q = 2$) to be most effective, closely followed by unigrams ($q = 1$). As with informers, we can also use hypernyms of regular words as SVM

[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

features (marked “Question bigrams + hypernyms” in Table 2).

3.2 Benefits from “perfect” informers

We first wished to test the hypothesis that identifying informer spans to an SVM learner can improve classification accuracy. Over and above the class labels, we had two volunteers tag the 6000 UIUC questions with informer spans (which we call “perfect”—agreement was near-perfect).

Features	Coarse	Fine
Question trigrams	91.2	77.6
All question $q$grams	87.2	71.8
All question unigrams	88.4	78.2
Question bigrams	91.6	79.4
+informer q-grams	94.0	82.4
+informer hypernyms	**94.2**	**88.0**
Question unigrams + all informer	93.4	88.0
Only informer	92.2	85.0
Question bigrams + hypernyms	91.6	79.4

Table 2: Percent accuracy with linear SVMs, “perfect” informer spans, and various feature encodings.

Observe in Table 2 that the unigram baseline is already quite competitive with the best prior numbers, and exploiting perfect informer spans beats all known numbers. It is clear that both *informer q-grams* and *informer hypernyms* are very valuable features for question classification. The fact that no improvement was obtained with over *Question bigrams* using *Question hypernyms* highlights the importance of choosing a few relevant tokens as informers and designing suitable features on them.

Table 3 (columns b and e) shows the benefits from perfect informers broken down into broad question types. Questions with *what* as the trigger are the biggest beneficiaries, and they also form by far the most frequent category.

The remaining question, one that we address in the rest of the paper, is whether we can effectively and accurately automate the process of providing informer spans to the question classifier.

3.3 Informers provided by heuristics

In §4 we will propose a non-trivial solution to the informer-tagging problem. Before that, we must justify that such machinery is indeed required.

Some leading QA systems extract words very similar in function to informers from the parse tree of the question. Some (Singhal et al., 2000) pick the head of the first noun phrase detected by a shallow parser, while others use the head of the noun phrase adjoining the main verb (Ramakrishnan et al., 2004). Yet others (Harabagiu et al., 2000; Hovy et al., 2001) use hundreds of (unpublished to our knowledge) hand-built pattern-matching rules on the output of a full-scale parser.

A natural baseline is to use these extracted words, which we call “heuristic informers”, with an SVM just like we used “perfect” informers. All that remains is to make the heuristics precise.

How: For questions starting with *how*, we use the bigram starting with *how* unless the next word is a verb.

Wh: If the wh-word is not *how*, *what* or *which*, use the wh-word in the question as a separate feature.

WhNP: For questions having *what* and *which*, use the WHNP if it encloses a noun. WHNP is the Noun Phrase corresponding to the Wh-word, given by a sentence parser (see §4.2).

NP1: Otherwise, for *what* and *which* questions, the first (leftmost) noun phrase is added to yet another feature subspace.

Table 3 (columns c and f) shows that these already-messy heuristic informers do not capture the same signal quality as “perfect” informers. Our findings corroborate Li and Roth (2002), who report little benefit from adding head chunk features for the fine classification task.

Moreover, observe that using heuristic informer features *without* any word features leads to rather poor performance (column c), unlike using perfect informers (column b) or even CRF-predicted informer (column d, see §4). These clearly establish that the notion of an informer is nontrivial.

4 Using CRFs to label informers

Given informers are useful but nontrivial to recognize, the next natural question is, how can we learn to identify them automatically? From earlier sections, it is clear (and we give evidence later, see Table 5) that sequence and syntax information will be

6 coarse classes

		B	Only Informers			B+	B+	B+
Type	#Quest.	(Bigrams)	Perf.Inf	H.Inf	CRF.Inf	Perf.Inf	H.Inf	CRF.Inf
what	349	**88.8**	89.4	69.6	79.3	**91.7**	87.4	91.4
which	11	**72.7**	100.0	45.4	81.8	**100.0**	63.6	81.8
when	28	100.0	100.0	100.0	100.0	100.0	100.0	100.0
where	27	100.0	96.3	100.0	96.3	100.0	100.0	100.0
who	47	100.0	100.0	100.0	100.0	100.0	100.0	100.0
how_*	32	100.0	96.9	100.0	100.0	100.0	100.0	100.0
rest	6	100.0	100.0	100.0	66.7	100.0	66.7	66.7
Total	500	91.6	92.2	*77.2*	84.6	94.2	*90.0*	93.4
50 fine classes								
what	349	**73.6**	82.2	61.9	78.0	**85.1**	79.1	83.1
which	11	**81.8**	90.9	45.4	73.1	**90.9**	54.5	81.8
when	28	100.0	100.0	100.0	100.0	100.0	100.0	100.0
where	27	92.6	85.2	92.6	88.9	88.9	92.5	88.9
who	47	97.9	93.6	93.6	93.6	100.0	100.0	97.9
how_*	32	87.5	84.3	81.2	78.1	87.5	90.6	90.6
rest	6	66.7	66.7	66.7	66.7	100.0	66.7	66.7
Total	500	79.4	85.0	*69.6*	78.0	88.0	*82.6*	86.2
		a	b	c	d	e	f	g

Table 3: Summary of % accuracy broken down by question type (referred from §3.2, §3.3 and §4.4). a: question bigrams, b: perfect informers only, c: heuristic informers only, d: CRF informers only, e–g: bigrams plus perfect, heuristic and CRF informers.

important.

We will model informer span identification as a sequence tagging problem. An automaton makes probabilistic transitions between hidden states y, one of which is an "informer generating state", and emits tokens x. We observe the tokens and have to guess which were produced from the "informer generating state".

Hidden Markov models are extremely popular for such applications, but recent work has shown that conditional random fields (CRFs) (Lafferty et al., 2001; Sha and Pereira, 2003) have a consistent advantage over traditional HMMs in the face of many redundant features. We refer the reader to the above references for a detailed treatment of CRFs. Here we will regard a CRF as largely a black box[3].

To train a CRF, we need a set of state nodes, a transition graph on these nodes, and tokenized text where each token is assigned a state. Once the CRF is trained, it can be applied to a token sequence, producing a predicted state sequence.

4.1 State transition models

We started with the common 2-state "in/out" model used in information extraction, shown in the left half of Figure 2. State "1" is the informer-generating state. Either state can be initial and final (double circle) states.

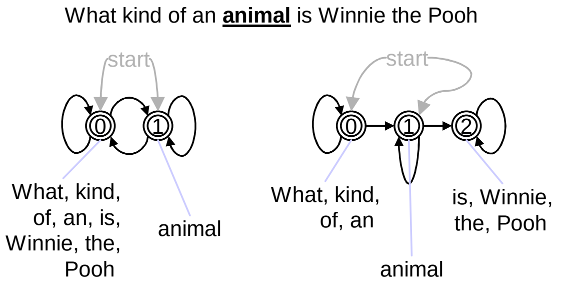
What kind of an **animal** is Winnie the Pooh
start
0
1
What, kind, of, an, is, Winnie, the, Pooh
animal
start
0
1
2
What, kind, of, an
is, Winnie, the, Pooh
animal

Figure 2: 2- and 3-state transition models.

The 2-state model can be myopic. Consider the question pair

[3] We used `http://crf.sourceforge.net/`

A: What country is the largest producer of wheat?
B: Name the largest producer of wheat

The $i \pm 1$ context of *producer* is identical in A and B. In B, for want of a better informer, we would want *producer* to be flagged as the informer, although it might refer to a country, person, animal, company, etc. But in A, *country* is far more precise.

Any 2-state model that depends on positions $i \pm 1$ to define features will fail to distinguish between A and B, and might select both *country* and *producer* in A. As we have seen with heuristic informers, polluting the informer pool can significantly hurt SVM accuracy.

Therefore we also use the 3-state "begin/in/out" (BIO) model. The initial state cannot be "2" in the 3-state model; all states can be final. The 3-state model allows at most one informer span. Once the 3-state model chooses *country* as the informer, it is unlikely to stretch state 1 up to *producer*.

There is no natural significance to using four or more states. Besides, longer range syntax dependencies are already largely captured by the parser.

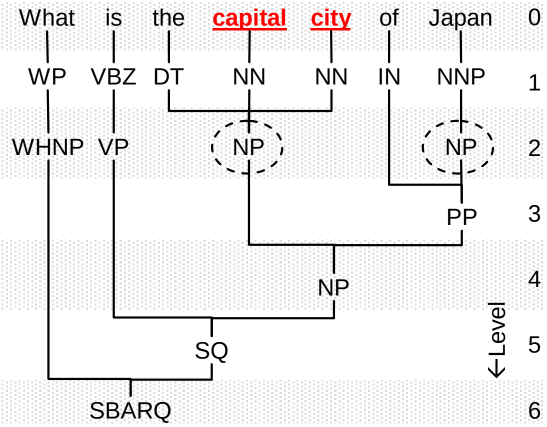

Figure 3: Stanford Parser output example.

4.2 Features from a parse of the question

Sentences with similar parse trees are likely to have the informer in similar positions. This was the intuition behind Zhang et al.'s tree kernel, and is also our starting point. We used the Stanford Lexicalized Parser (Klein and Manning, 2003) to parse the question. (We assume familiarity with parse tree notation for lack of space.) Figure 3 shows a sample parse tree organized in levels. Our first step was to translate the parse tree into an equivalent multi-resolution tabular format shown in Table 4.

i	1	2	3	4	5	6	7
y_i	0	0	0	1	1	2	2
x_i	What	is	the	capital	city	of	Japan
$\ell \downarrow$	Features for x_is						
1	WP,1	VBZ,1	DT,1	NN,1	NN,1	IN,1	NNP,1
2	WHNP,1	VP,1	NP,1	**NP**,1	NP,1	Null,1	**NP,2**
3	Null,1	Null,1	Null,1	Null,1	Null,1	PP,1	PP,1
4	Null,1	Null,1	NP,1	NP,1	NP,1	NP,1	NP,1
5	Null,1	SQ,1	SQ,1	SQ,1	SQ,1	SQ,1	SQ,1
6	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ

Table 4: A multi-resolution tabular view of the question parse showing `tag` and `num` attributes. *capital city* is the informer span with $y = 1$.

Cells and attributes: A labeled question comprises the token sequence $x_i; i = 1, \ldots$ and the label sequence $y_i, i = 1, \ldots$ Each x_i leads to a column vector of observations. Therefore we use matrix notation to write down x: A table cell is addressed as $x[i, \ell]$ where i is the token position (column index) and ℓ is the level or row index, 1–6 in this example. (Although the parse tree can be arbitrarily deep, we found that using features from up to level $\ell = 2$ was adequate.)

Intuitively, much of the information required for spotting an informer can be obtained from the part of speech of the tokens and phrase/clause attachment information. Conversely, specific word information is generally sparse and misleading; the same word may or may not be an informer depending on its position. E.g., "What birds eat snakes?" and "What snakes eat birds?" have the same words but different informers. Accordingly, we observe two properties at each cell:

`tag`: The syntactic class assigned to the cell by the parser, e.g. $x[4, 2]$`.tag` = `NP`. It is well-known that POS and chunk information are major clues to informer-tagging, specifically, informers are often nouns or noun phrases.

`num`: Many heuristics exploit the fact that the first NP is known to have a higher chance of containing informers than subsequent NPs. To capture this positional information, we define `num` of a cell at $[i, \ell]$ as one plus the number of distinct contiguous chunks to the left of $[i, \ell]$ with `tag`s equal to $x[4, 2]$`.tag`. E.g., at level 2 in the table above, *the capital city* forms the first NP, while *Japan* forms the second NP. Therefore $x[7,2]\texttt{.num} = 2$.

In conditional models, it is notationally convenient to express features as functions on (x_i, y_i). To one unfamiliar with CRFs, it may seem strange that y_i is passed as an argument to features. At training time, y_i is indeed known, and at testing time, the CRF algorithm efficiently finds the most probable sequence of y_is using a Viterbi search. True labels are not revealed to the CRF at testing time.

Cell features `IsTag` and `IsNum`: E.g., the observation "$y_4 = 1$ and $x[4,2]\texttt{.tag} = \texttt{NP}$" is captured by the statement that "position 4 fires the feature $\texttt{IsTag}_{1,\texttt{NP},2}$" (which has a boolean value). There is an $\texttt{IsTag}_{y,t,\ell}$ feature for each (y,t,ℓ) triplet. Similarly, for every possible state y, every possible `num` value n (up to some maximum horizon), and every level ℓ, we define boolean features $\texttt{IsNum}_{y,n,\ell}$. E.g., position 7 fires the feature $\texttt{IsNum}_{2,2,2}$ in the 3-state model, capturing the statement "$x[7,2]\texttt{.num} = 2$ and $y_7 = 2$".

Adjacent cell features `IsPrevTag` and `IsNextTag`: Context can be exploited by a CRF by coupling the state at position i with observations at positions adjacent to position i (extending to larger windows did not help). To capture this, we use more boolean features: position 4 fires the feature $\texttt{IsPrevTag}_{1,\texttt{DT},1}$ because $x[3,1]\texttt{.tag} = \texttt{DT}$ and $y_4 = 1$. Position 4 also fires $\texttt{IsPrevTag}_{1,\texttt{NP},2}$ because $x[3,2]\texttt{.tag} = \texttt{NP}$ and $y_4 = 1$. Similarly we define a $\texttt{IsNextTag}_{y,t,\ell}$ feature for each possible (y,t,ℓ) triple.

State transition features `IsEdge`: Position i fires feature $\texttt{IsEdge}_{u,v}$ if $y_{i-1} = u$ and $y_i = v$. There is one such feature for each state-pair (u,v) allowed by the transition graph. In addition we have sentinel features $\texttt{IsBegin}_u$ and $\texttt{IsEnd}_u$ marking the beginning and end of the token sequence.

4.3 Informer-tagging accuracy

We study the accuracy of our CRF-based informer tagger wrt human informer annotations. In the next section we will see the effect of CRF tagging on question classification.

There are at least two useful measures of informer-tagging accuracy. Each question has a known set I_k of informer tokens, and gets a set of tokens I_c flagged as informers by the CRF. For each question, we can grant ourself a reward of 1 if $I_c = I_k$, and 0 otherwise. In §3.1, informers were regarded as a separate (high-value) bag of words. Therefore, overlap between I_c and I_k would be a reasonable predictor of question classification accuracy. We use the Jaccard similarity $|I_k \cap I_c| / |I_k \cup I_c|$. Table 5 shows the effect of using diverse feature sets.

	Fraction	Jaccard
Features used	$I_c = I_k$	overlap
IsTag	0.368	0.396
+IsNum	0.474	0.542
+IsPrevTag+IsNextTag	0.692	0.751
+IsEdge+IsBegin+IsEnd	**0.848**	**0.867**

Table 5: Effect of feature choices.

- `IsTag` features are not adequate.
- `IsNum` features improve accuracy 10–20%.
- `IsPrevTag` and `IsNextTag` ("+Prev +Next") add over 20% of accuracy.
- `IsEdge` transition features help exploit Markovian dependencies and adds another 10–15% accuracy, showing that sequential models are indeed required.

Type	#Quest.	Heuristic	2-state	3-state
		Informers	CRF	CRF
what	349	57.3	68.2	83.4
which	11	77.3	83.3	77.2
when	28	75.0	98.8	100.0
where	27	84.3	100.0	96.3
who	47	55.0	47.2	96.8
how_*	32	90.6	88.5	93.8
rest	6	66.7	66.7	77.8
Total	500	62.4	71.2	86.7

Table 6: Effect of number of CRF states, and comparison with the heuristic baseline (Jaccard accuracy expressed as %).

Table 6 shows that the 3-state CRF performs much better than the 2-state CRF, especially on difficult questions with *what* and *which*. It also compares the Jaccard accuracy of informers found by the CRF vs. informers found by the heuristics described in §3.3. Again we see a clear superiority of the CRF approach.

Unlike the heuristic approach, the CRF approach is relatively robust to the parser emitting a somewhat incorrect parse tree, which is not uncommon. The heuristic approach picks the “easy” informer, *who*, over the better one, *CEO*, in “Who is the CEO of IBM”. Its bias toward the NP-head can also be a problem, as in “What country’s *president* . . .”.

4.4 Question classification accuracy

We have already seen in §3.2 that perfect knowledge of informers can be a big help. Because the CRF can make mistakes, the margin may decrease. In this section we study this issue.

We used questions with human-tagged informers (§3.2) to train a CRF. The CRF was applied back on the training questions to get informer predictions, which were used to train the 1-vs-1 SVM meta-learner (§3). Using CRF-tagged and not human-tagged informers may seem odd, but this lets the SVM learn and work around systematic errors in CRF outputs.

Results are shown in columns d and g of Table 3. Despite the CRF tagger having about 15% error, we obtained 86.2% SVM accuracy which is rather close to the the SVM accuracy of 88% with perfect informers.

The CRF-generated tags, being on the training data, might be more accurate that would be for unseen test cases, potentially misleading the SVM. This turns out not to be a problem: clearly we are very close to the upper bound of 88%. In fact, anecdotal evidence suggests that using CRF-assigned tags actually helped the SVM.

5 Conclusion

We presented a new approach to inferring the type of the answer sought by a well-formed natural language question. We introduced the notion of a span of *informer tokens* and extract it using a sequential graphical model with a novel feature representation derived from the parse tree of the question. Our approach beats the accuracy of recent algorithms, even ones that used max-margin methods with sophisticated kernels defined on parse trees.

An intriguing feature of our approach is that when an informer (*actor*) is narrower than the question class (*person*), we can exploit direct hypernymy connections like *actor* to *Tom Hanks*, if available. Existing knowledge bases like WordNet and Wikipedia, combined with intense recent work (Etzioni et al., 2004) on bootstrapping is-a hierarchies, can thus lead to potentially large benefits.

Acknowledgments: Thanks to Sunita Sarawagi for help with CRFs, and the reviewers for improving the presentation.

References

P. K Chan and S. J Stolfo. 1993. Experiments in multistrategy learning by meta-learning. In *CIKM*, pages 314–323, Washington, DC.

S Dumais, M Banko, E Brill, J Lin, and A Ng. 2002. Web question answering: Is more always better? In *SIGIR*, pages 291–298.

O Etzioni, M Cafarella, et al. 2004. Web-scale information extraction in KnowItAll. In *WWW Conference*, New York. ACM.

K Hacioglu and W Ward. 2003. Question classification with support vector machines and error correcting codes. In *HLT*, pages 28–30.

S Harabagiu, D Moldovan, M Pasca, R Mihalcea, M Surdeanu, R Bunescu, R Girju, V Rus, and P Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *TREC 9*, pages 479–488. NIST.

E Hovy, L Gerber, U Hermjakob, M Junk, and C.-Y Lin. 2001. Question answering in Webclopedia. In *TREC 9*. NIST.

R Khardon, D Roth, and L. G Valiant. 1999. Relational learning for NLP using linear threshold elements. In *IJCAI*.

D Klein and C. D Manning. 2003. Accurate unlexicalized parsing. In *ACL*, volume 41, pages 423–430.

C Kwok, O Etzioni, and D. S Weld. 2001. Scaling question answering to the Web. In *WWW Conference*, volume 10, pages 150–161, Hong Kong.

J Lafferty, A McCallum, and F Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

X Li and D Roth. 2002. Learning question classifiers. In *COLING*, pages 556–562.

G Ramakrishnan, S Chakrabarti, D. A Paranjpe, and P Bhattacharyya. 2004. Is question answering an acquired skill? In *WWW Conference*, pages 111–120, New York.

F Sha and F Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*, pages 134–141.

A Singhal, S Abney, M Bacchiani, M Collins, D Hindle, and F Pereira. 2000. AT&T at TREC-8. In *TREC 8*, pages 317–330. NIST.

D Zhang and W Lee. 2003. Question classification using support vector machines. In *SIGIR*, pages 26–32.

Z Zheng. 2002. AnswerBus question answering system. In *HLT*.