Answer Extraction as Sequence Tagging with Tree Edit Distance

Xuchen Yao and **Benjamin Van Durme**
Johns Hopkins University
Baltimore, MD, USA

Chris Callison-Burch*
University of Pennsylvania
Philadelphia, PA, USA

Peter Clark
Vulcan Inc.
Seattle, WA, USA

Abstract

Our goal is to extract answers from pre-retrieved sentences for Question Answering (QA). We construct a linear-chain Conditional Random Field based on pairs of questions and their possible answer sentences, learning the association between questions and answer types. This casts answer extraction as an answer sequence tagging problem for the first time, where knowledge of shared structure between question and source sentence is incorporated through features based on Tree Edit Distance (TED). Our model is free of manually created question and answer templates, fast to run (processing 200 QA pairs per second excluding parsing time), and yields an F1 of 63.3% on a new public dataset based on prior TREC QA evaluations. The developed system is open-source, and includes an implementation of the TED model that is state of the art in the task of ranking QA pairs.

1 Introduction

The success of IBM's Watson system for Question Answering (QA) (Ferrucci et al., 2010) has illustrated a continued public interest in this topic. Watson is a sophisticated piece of software engineering consisting of many components tied together in a large parallel architecture. It took many researchers working full time for years to construct. Such resources are not available to individual academic researchers. If they are interested in evaluating new ideas on some aspect of QA, they must either construct a full system, or create a focused subtask paired with a representative dataset. We follow the latter approach and focus on the task of answer extraction, i.e., producing the exact answer strings for a question.

We propose the use of a linear-chain Conditional Random Field (CRF) (Lafferty et al., 2001) in order to cast the problem as one of *sequence tagging* by labeling each token in a candidate sentence as either Beginning, Inside or Outside (BIO) of an answer. This is to our knowledge the first time a CRF has been used to extract answers.[1] We utilize not only traditional contextual features based on POS tagging, dependency parsing and Named Entity Recognition (NER), but most importantly, features extracted from a Tree Edit Distance (TED) model for aligning an answer sentence tree with the question tree. The linear-chain CRF, when trained to learn the associations between question and answer types, is a robust approach against error propagation introduced in the NLP pipeline. For instance, given an NER tool that always (i.e., in both training and test data) recognizes the pesticide DDT as an ORG, our model realizes, when a question is asked about the type of chemicals, the correct answer might be *incorrectly but consistently* recognized as ORG by NER. This helps reduce errors introduced by wrong answer types, which were estimated as the most significant contributer (36.4%) of errors in the then state-of-the-art QA system of Moldovan et al. (2003).

The features based on TED allow us to draw the

*Performed while faculty at Johns Hopkins University.

[1] CRFs have been used in judging answer-bearing sentences (Shima et al., 2008; Ding et al., 2008; Wang and Manning, 2010), but not extracting exact answers from these sentences.

connection between the question and answer sentences *before* answer extraction, whereas traditionally the exercise of *answer validation* (Magnini et al., 2002; Penas et al., 2008; Rodrigo et al., 2009) has been performed *after* as a remedy to ensure the answer is really "about" the question.

Motivated by a desire for a fast runtime,[2] we base our TED implementation on the dynamic-programming approach of Zhang and Shasha (1989), which helps our final system process 200 QA pairs per second on standard desktop hardware, when input is syntactically pre-parsed.

In the following we first provide background on the TED model, going on to evaluate our implementation against prior work in the context of question answer sentence ranking (QASR), achieving state of the art in that task. We then describe how we couple TED features to a linear-chain CRF for answer extraction, providing the set of features used, and finally experimental results on an extraction dataset we make public (together with the software) to the community.[3] Related prior work is interspersed throughout the paper.

2 Tree Edit Distance Model

Tree Edit Distance (§2.1) models have been shown effective in a variety of applications, including textual entailment, paraphrase identification, answer ranking and information retrieval (Reis et al., 2004; Kouylekov and Magnini, 2005; Heilman and Smith, 2010; Augsten et al., 2010). We chose the variant proposed by Heilman and Smith (2010), inspired by its simplicity, generality, and effectiveness. Our approach differs from those authors in their reliance on a greedy search routine to make use of a complex tree kernel. With speed a consideration, we opted for the dynamic-programming solution of Zhang and Shasha (1989) (§2.1). We added new lexical-semantic features §(2.2) to the model and then evaluated our implementation on the QASR task, showing strong results §(2.3).

Feature	Description
distance	tree edit distance from answer sentence to question
renNoun renVerb renOther	# edits changing POS from or to noun, verb, or other types
insN, insV, insPunc, insDet, insOtherPos	# edits inserting a noun, verb, punctuation mark, determiner or other POS types
delN, delV, ...	deletion mirror of above
ins{N,V,P}Mod insSub, insObj insOtherRel	# edits inserting a modifier for {noun, verb, preposition}, subject, object or other relations
delNMod, ...	deletion mirror of above
renNMod, ...	rename mirror of above
XEdits	# basic edits plus sum of ins/del/ren edits
alignNodes, alignNum, alignN, alignV, alignProper	# aligned nodes, and those that are numbers, nouns, verbs, or proper nouns

Table 1: Features for ranking QA pairs.

2.1 Cost Design and Edit Search

Following Bille (2005), we define an *edit script* between trees T_1, T_2 as the edit sequence transforming T_1 to T_2 according to a *cost function*, with the total summed cost known as the *tree edit distance*. *Basic edit* operations include: *insert*, *delete* and *rename*.

With T a dependency tree, we represent each node by three fields: lemma, POS and the type of dependency relation to the node's parent (DEP). For instance, Mary/nnp/sub is the proper noun *Mary* in subject position.

Basic edits are refined into 9 types, where the first six (INS_LEAF, INS_SUBTREE, INS, DEL_LEAF, DEL_SUBTREE, DEL) insert or delete a leaf node, a whole subtree, or a node that is neither a leaf nor part of a whole inserted subtree. The last three (REN_POS, REN_DEP, REN_POS_DEP) serve to rename a POS tag, dependency relation, or both.

[2] For instance, Watson was designed under the constraint of a 3 second response time, arising from its intended live use in the television gameshow, `Jeopardy!`.

[3] `http://code.google.com/p/jacana/`

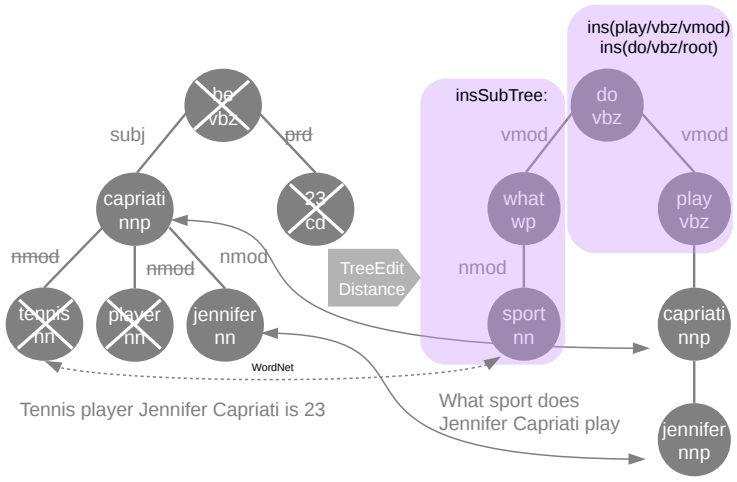

Figure 1: Edits transforming a source sentence (left) to a question (right). Each node consists of: lemma, POS tag and dependency relation, with root nodes and punctuation not shown. Shown includes deletion ($\times$ and ~~strikethrough~~ on the left), alignment (arrows) and insertion (shaded area). Order of operations is not displayed. The standard TED model does not capture the alignment between *tennis* and *sport* (see Section 2.2).

We begin by uniformly assigning basic edits a cost of 1.0,[4] which brings the cost of a full node insertion or deletion to 3 (all the three fields inserted or deleted). We allow renaming of POS and/or relation type iff the lemmas of source and target nodes are identical.[5] When two nodes are identical and thus do not appear in the edit script, or when two nodes are renamed due to the same lemma, we say they are *aligned* by the tree edit model (see Figure 1).

We used Zhang and Shasha (1989)'s dynamic programming algorithm to produce an optimal edit script with the lowest tree edit distance. The approach explores both trees in a bottom-up, post-order manner, running in time:

$$O(|T_1|\,|T_2|\min(D_1, L_1)\min(D_2, L_2))$$

where $|T_i|$ is the number of nodes, D_i is the depth, and L_i is the number of leaves, with respect to tree T_i.

Additionally, we fix the cost of stopword renaming to 2.5, even in the case of identity, regardless of whether two stopwords have the same POS tags or relations. Stopwords tend to have fixed POS tags and dependency relations, which often leads to less expensive alignments as compared to renaming content terms. In practice this gave stopwords "too much say" in guiding the overall edit sequence.

The resultant system is fast in practice, processing 10,000 pre-parsed tree pairs per second on a contemporary machine.[6]

2.2 TED for Sentence Ranking

The task of Question Answer Sentence Ranking (QASR) takes a question and a set of source sentences, returning a list sorted by the probability likelihood that each sentence contains an appropriate answer. Prior work in this includes that of: Punyakanok et al. (2004), based on mapping syntactic dependency trees; Wang et al. (2007) utilizing Quasi-Synchronous Grammar (Smith and Eisner, 2006); Heilman and Smith (2010) using TED; and Shima et al. (2008), Ding et al. (2008) and Wang and Manning (2010), who each employed a CRF in various ways. Wang et al. (2007) made their dataset public, which we use here for system validation. To date, models based on TED have shown the best performance for this task.

Our implementation follows Heilman and Smith (2010), with the addition of 15 new features beyond their original 33 (see Table 1). Based on results

[4]This applies separately to each element of the tripartite structure; e.g., deleting a POS entry, inserting a lemma, etc.

[5]This is aimed at minimizing node variations introduced by morphology differences, tagging or parsing errors.

[6]In later tasks, feature extraction and decoding will slow down the system, but the final system was still able to process 200 pairs per second.

set	source	#ques.	#pairs	%pos.	len.
TRAIN-ALL	TREC8-12	1229	53417	12.0	any
TRAIN	TREC8-12	94	4718	7.4	≤ 40
DEV	TREC13	82	1148	19.3	≤ 40
TEST	TREC13	89	1517	18.7	≤ 40

Table 2: Distribution of data, with imbalance towards negative examples (sentences without an answer).

System	MAP	MRR
Wang et al. (2007)	0.6029	0.6852
Heilman and Smith (2010)	0.6091	0.6917
Wang and Manning (2010)	0.5951	0.6951
this paper (48 features)	0.6319	0.7270
+WNsearch	**0.6371**	0.7301
+WNfeature (11 more feat.)	0.6307	**0.7477**

Table 3: Results on the QA Sentence Ranking task.

in DEV, we extract edits in the direction from the source sentence to the question.

In addition to syntactic features, we incorporated the following lexical-semantic relations from WordNet: *hypernym* and *synonym* (nouns and verbs); *entailment* and *causing* (verbs); and *membersOf*, *substancesOf*, *partsOf*, *haveMember*, *haveSubstance*, *havePart* (nouns). Such relations have been used in prior approaches to this task (Wang et al., 2007; Wang and Manning, 2010), but not in conjunction with the model of Heilman and Smith (2010).

These were made into features in two ways: **WNsearch** loosens renaming and alignment within the TED model from requiring strict lemma equality to allowing lemmas that shared any of the above relations, leading to renaming operations such as REN_...(`country`, `china`) and REN_...(`sport`, `tennis`); **WNfeature** counts how many words between the sentence and answer sentence have each of the above relations, separately as 10 independent features, plus an aggregate count for a total of 11 new features beyond the earlier 48.

These features were then used to train a logistic regression model using Weka (Hall et al., 2009).

2.3 QA Sentence Ranking Experiment

We trained and tested on the dataset from Wang et al. (2007), which spans QA pairs from TREC QA 8-13 (see Table 2). Per question, sentences with non-stopword overlap were first retrieved from the task collection, which were then compared against the TREC answer pattern (in the form of Perl regular expressions). If a sentence matched, then it was deemed a (noisy) positive example. Finally, TRAIN, DEV and TEST were manually corrected for errors. Those authors decided to limit candidate source sentences to be no longer than 40 words.[7] Keeping with prior work, those questions with only positive or negative examples were removed, leaving 94 of the original 100 questions for evaluation.

The data was processed by Wang et al. (2007) with the following tool chain: POS tags via MXPOST (Ratnaparkhi, 1996); parse trees via MSTParser (McDonald et al., 2005) with 12 coarse-grained dependency relation labels; and named entities via Identifinder (Bikel et al., 1999). Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are reported in Table 3. Our implementation gives state of the art performance, and is furthered improved by our inclusion of semantic features drawn from WordNet.[8]

3 CRF with TED for Answer Extraction

In this section we move from *ranking* source sentences, to the next QA stage: *answer extraction*. Given our competitive TED-based alignment model, the most obvious solution to extraction would be to report those spans aligned from a source sentence to a question's *wh-* terms. However, we show that this approach is better formulated as a (strongly indicative) feature of a larger set of answer extraction signals.

3.1 Sequence Model

Figure 2 illustrates the task of tagging each token in a candidate sentence with one of the following la-

[7] TRAIN-ALL is not used in QASR, but later for answer extraction; TRAIN comes from the first 100 questions of TRAIN-ALL.

[8] As the test set is of limited size (94 questions), then while our MAP/MRR scores are $2.8\% \sim 5.6\%$ higher than prior work, this is not statistically significant according to the Paired Randomization Test (Smucker et al., 2007), and thus should be considered *on par* with the current state of the art.

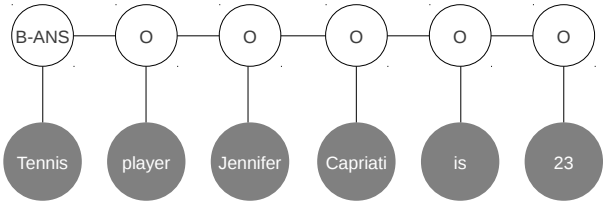

Figure 2: An example of linear-chain CRF for answer sequence tagging.

bels: B-ANSWER (beginning of answer), I-ANSWER (inside of answer), O (outside of answer).

Besides local POS/NER/DEP features, at each token we need to inspect the entire input to connect the answer sentence with the question sentence through tree edits, drawing features from the question and the edit script, motivating the use of a linear-chain CRF model (Lafferty et al., 2001) over HMMs. To the best of our knowledge this is the first time a CRF has been used to label answer fragments, despite success in other sequence tagging tasks.

3.2 Feature Design

In this subsection we describe the local and global features used by the CRF.

Chunking We use the POS/NER/DEP tags directly just as one would in a chunking task. Specifically, suppose t represents the current token position and $pos[t]$ its POS tag, we extract unigram, bigram and trigram features over the local context, e.g., $pos[t-2], pos[t-2] : pos[t-1]$, and $pos[t-2] : pos[t-1] : pos[t]$. Similar features are extracted for named entity types ($ner[t]$), and dependency relation labels ($dep[t]$).

Our intuition is these chunking features should allow for learning which types of words tend to be answers. For instance, we expect adverbs to be assigned lower feature weights as they are rarely a part of answer, while prepositions may have different feature weights depending on their context. For instance, *of* in *kind of silly* has an adjective on the right, and is unlikely to be the Beginning of an answer to a TREC-style question, as compared to *in* when paired with a question on time, such as seen in an answer *in 90 days*, where the preposition is followed by a number then a noun.

Feature	Description
edit=X	type of edit feature. X: DEL, DEL_SUBTREE, DEL_LEAF, REN_POS, REN_DEP, REN_POS_DEP or ALIGN.
X_pos=? X_ner=? X_dep=?	Delete features. X is either DEL, DEL_SUBTREE or DEL_LEAF. ? represents the corresponding POS/NER/DEP of the current token.
Xpos_from=?$_f$ Xpos_to=?$_t$ Xpos_f_t=?$_f$_?$_t$ Xner_from=?$_f$ Xner_to=?$_t$ Xner_f_t=?$_f$_?$_t$ Xdep_from=?$_f$ Xdep_to=?$_t$ Xdep_f_t=?$_f$_?$_t$	Rename features. X is either REN_POS, REN_DEP or REN_POS_DEP. Suppose word f in answer is renamed to word t in question, then ?$_f$ and ?$_t$ represent corresponding POS/NER/DEP of f and t.
align_pos=? align_ner=? align_dep=?	Align features. ? represents the corresponding POS/NER/DEP of the current token.

Table 4: Features based on edit script for answer sequence tagging.

Question-type Chunking features do not capture the connection between question word and answer types. Thus they have to be combined with question types. For instance, how many questions are usually associated with numeric answer types. We encode each major question-type: who, whom, when, where, how many, how much, how long, and then for each token, we combine the question term with its chunking features described in (most tokens have different features because they have different POS/NER/DEP types). One feature example of the QA pair how_much/100_dollars for the word 100 would be: qword=how_much|pos[t]=CD|pos[t+1]=NNS. We expect high weight for this feature since it is a good pattern for matching question type and answer type. Similar features also apply to what, which, why and how questions, even though they do not indicate an answer type as clearly as how much does.

Some extra features are designed for what/which questions per required answer types. The question

dependency tree is analyzed and the Lexical Answer Type (LAT) is extracted. The following are some examples of LAT for what questions:

- color: what is Crips' gang color?
- animal: what kind of animal is an agouti?

The extra LAT=? feature is also used with chunking features for what/which questions.

There is significant prior work in building specialized templates or classifiers for labeling question types (Hermjakob, 2001; Li and Roth, 2002; Zhang and Lee, 2003; Hacioglu and Ward, 2003; Metzler and Croft, 2005; Blunsom et al., 2006; Moschitti et al., 2007). We designed our shallow question type features based on the intuitions of these prior work, with the goal of having a relatively compact approach that still extracts useful predictive signal. One possible drawback, however, is that if an LAT is not observed during training but shows up in testing, the sequence tagger would not know which answer type to associate with the question. In this case it falls back to the more general qword=? feature and will most likely pick the type of answers that are mostly associated with what questions in training.

Edit script Our TED module produces an edit trace for each word in a candidate sentence: the word is either deleted, renamed (if there is a word of the same lemma in the question tree) or strictly aligned (if there is an identical node in the question tree). A word in the deleted edit sequence is a cue that it could be the answer. A word being aligned suggests it is less likely to be an answer. Thus for each word we extract features based on its edit type, shown in Table 4.

These features are also appended with the token's POS/NER/DEP information. For instance, a deleted noun usually carries higher edit feature weights than an aligned adjective.

Alignment distance We observed that a candidate answer often appears close to an aligned word (i.e., answer tokens tend to be located "nearby" portions of text that align across the pair), especially in compound noun constructions, restrictive clauses, preposition phrases, etc. For instance, in the following pair, the answer Limp Bizkit comes from the leading compound noun:

- What is the name of Durst 's group?
- Limp Bizkit lead singer Fred Durst did a lot ...

Past work has designed large numbers of specific templates aimed at these constructions (Soubbotin, 2001; Ravichandran et al., 2003; Clark et al., 2003; Sneiders, 2002). Here we use a single general feature that we expect to pick up much of this signal, without the significant feature engineering.

Thus we incorporated a simple feature to *roughly* model this phenomenon. It is defined as the distance to the nearest aligned nonstop word in the original word order. In the above example, the only aligned nonstop word is Durst. Then this nearest alignment distance feature for the word Limp is:

nearest_dist_to_align(Limp):5

This is the only integer-valued feature. All other features are binary-valued. Note this feature does not specify answer types: an adverb close to an aligned word can also be wrongly taken as a strong candidate. Thus we also include a version of the POS/NER/DEP based feature for each token:

- nearest_dist_pos(Limp)=NNP
- nearest_dist_dep(Limp)=NMOD
- nearest_dist_ner(Limp)=B-PERSON

3.3 Overproduce-and-vote

We make an assumption that each sentence produces a candidate answer and then vote among all answer candidates to select the most-voted as the answer to the original question. Specifically, this overproduce-and-vote strategy applies voting in two places:

1. If there are overlaps between two answer candidates, a partial vote is performed. For instance, for a when question, if one answer candidate is April , 1994 and the other is 1994, then besides the base vote of 1, both candidates have an extra partial vote of $\#_{\text{overlap}}/\#_{\text{total words}} = 1/4$. We call this *adjusted vote*.

2. If the CRF fails to find an answer, we still try to "force" an answer out of the tagged sequence, O's). thus *forced vote*. Due to its lower credibility (the sequence tagger does not think it is an answer), we manually downweight the prediction score by a factor of 0.1 (divide by 10).

```
During what war did Nimitz serve ?
O      O:0.921060      Conant
O      O:0.991168      had
O      O:0.997307      been
O      O:0.998570      a
O      O:0.998608      photographer
O      O:0.999005      for
O      O:0.877619      Adm
O      O:0.988293      .
O      O:0.874101      Chester
O      O:0.924568      Nimitz
O      O:0.970045      during
B–ANS  O:0.464799      World
I–ANS  O:0.493715      War
I–ANS  O:0.449017      II
O      O:0.915448      .
```

Figure 3: A sample sequence tagging output that fails to predict an answer. From line 2 on, the first column is the reference output and the second column is the model output with the marginal probability for predicated labels. Note that World War II has much lower probabilities as an O than others.

The modified score for an answer candidate is thus: total vote = adjusted vote + $0.1 \times$ forced vote. To compute forced vote, we make the following observation. Sometimes the sequence tagger does not tag an answer in a candidate sentence at all, if there is not enough probability mass accumulated for B-ANS. However, a possible answer can still be caught if it has an "outlier" marginal probability. Figure 3 shows an example. The answer candidate World War II has a much lower marginal probability as an "O" but still not low enough to be part of B-ANS/I-ANS.

To catch such an outlier, we use Median Absolute Deviation (MAD), which is the median of the absolute deviation from the median of a data sequence. Given a data sequence $\mathbf{x}$, MAD is defined as:

$$\text{MAD}(\mathbf{x}) = \text{median}(\mid \mathbf{x} - \text{median}(\mathbf{x}) \mid)$$

Compared to mean value and standard deviation, MAD is more robust against the influence of outliers since it does not directly depend on them. We select those words whose marginal probability is 50 times of MAD away from the median of the whole sequence as answer candidates. They contribute to the forced vote. Downweight ratio (0.1) and MAD ratio (50) were hand-tuned on DEV.[9]

System	Train	Prec.%	Rec.%	F1%
CRF	TRAIN	55.7	43.8	49.1
	TRAIN-ALL	**67.2**	50.6	57.7
CRF +WNsearch	TRAIN	58.6	46.1	51.6
	TRAIN-ALL	66.7	49.4	56.8
CRF forced	TRAIN	54.5	53.9	54.2
	TRAIN-ALL	60.9	59.6	60.2
CRF forced +WNsearch	TRAIN	55.2	53.9	54.5
	TRAIN-ALL	63.6	**62.9**	**63.3**

Table 5: Performance on TEST. "CRF" only takes votes from candidates tagged by the sequence tagger. "CRF forced" (described in §3.3) further collects answer candidates from sentences that CRF does not tag an answer by detecting outliers.

4 Experiments

4.1 QA Results

The dataset listed in Table 2 was not designed to include an answer for each positive answer sentence, but only a binary indicator on whether a sentence contains an answer. We used the answer pattern files (in Perl regular expressions) released along with TREC8-13 to pinpoint the exact answer fragments. Then we manually checked TRAIN, DEV, and TEST for errors. TRAIN-ALL already came as a noisy dataset so we did not manually clean it, also due to its large size.

We trained on only the positive examples of TRAIN and TRAIN-ALL separately with CRFsuite (Okazaki, 2007). The reason for training solely with positive examples is that they only constitute 10% of all training data and if trained on all, the CRF tagger was very biased on negative examples and reluctant to give an answer for most of the questions. The CRF tagger attempted an answer for about $2/3$ of all questions when training on just positive examples.

DEV was used to help design features. A practical benefit of our compact approach is that an entire round of feature extraction, training on TRAIN and testing on DEV took less than one minute. Table 5

[9] One might further improve this by leveraging the probability of a sentence containing an answer from the QA pair ranker described in Section 2 or via the conditional probability of the sequence labels, $p(\mathbf{y} \mid \mathbf{x})$, under the CRF.

reports F1 scores on both the positive and negative examples of TEST.

Our baseline model, which aligns the question word with some content word in the answer sentence,[10] achieves 31.4% in F1. This model does not require any training. "CRF" only takes votes from those sentences with an identified answer. It has the best precision among all models. "CRF forced" also detects outliers from sentences not tagged with an answer. Large amount of training data, even noisy, is helpful. In general TRAIN-ALL is able to boost the F1 value by 7 $\sim$ 8%. Also, the overgenerate-and-vote strategy, used by the "forced" approach, greatly increased recall and achieved the best F1 value.

We also experimented with the two methods utilizing WordNet in Section 2.2 , i.e., WNsearch and WNfeature. In general, WNsearch helps F1 and yields the best score (63.3%) for this task. For WNfeature[11] we observed that the CRF model converged to a larger objective likelihood with these features. However, it did not make a difference in F1 after overgenerate-and-vote.

Finally, we found it difficult to do a head-to-head comparison with other QA systems on this task.[12] Thus we contribute this dataset to the community, hoping to solicit direct comparisons in the future. Also, we believe our chunking and question-type features capture many intuitions most current QA systems rely on, while our novel features are based on TED. We further conduct an ablation test to compare traditional and new QA features.

4.2 Ablation Test

We did an ablation test for each of the four types of features. Note that the question type features are used in combination with chunking features (e.g., qword=how_much|pos[t]=CD|pos[t+1]=NN), while the chunking feature is defined over POS/NER/DEP separately. We tested the CRF model with deletion of one of the following features each time:

- POS, NER or DEP. These features are all combined with question types.
- The three of the above. Deletion of these features also deletes question type feature implicitly.
- EDIT. Features extracted from edit script.
- ALIGN. Alignment distance features.
- The two of the above, based on the TED model.

	CRF	Forced		CRF	Forced
All	49.1	54.2	-above 3	19.4	25.3
-POS	44.7	48.9	-EDIT	44.3	47.5
-NER	44.0	50.8	-ALIGN	47.4	51.1
-DEP	49.4	54.5	-above 2	40.5	42.0

Table 6: F1 based on feature ablation tests.

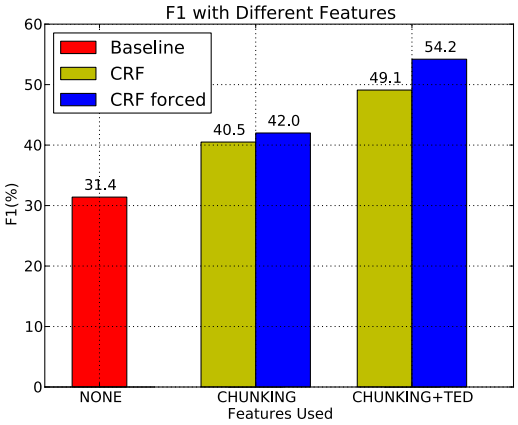

Figure 4: Impact of adding features based on chunking and question-type (CHUNKING) and tree edits (TED), e.g., EDIT and ALIGN.

Table 6 shows the F1 scores of ablation test when trained on TRAIN. NER and EDIT are the two single most significant features. NER is important because it closely relates question types with answer entity types (e.g., qword=who|ner[t]=PERSON). EDIT is also important because it captures the syntactic association between question tree and answer tree. Taking out all three POS/NER/DEP features means the chunking and question type features do not fire anymore. This has the biggest impact on F1. Note the feature redundancy here: the question type features are combined with all three POS/NER/DEP features

[10] This only requires minimal modification to the original TED algorithm: the question word is aligned with a certain word in the answer tree instead of being inserted. Then the whole subtree headed by the aligned word counts as the answer.

[11] These are binary features indicating whether an answer candidate has a WordNet relation (c.f. §2.2) with the LAT. For instance, tennis is a hyponym of the LAT word sport in the what sport question in Figure 1.

[12] Reasons include: most available QA systems either retrieve sentences from the web, have different preprocessing steps, or even include templates learned from our test set.

thus taking out a single one does not decrease performance much. However, since TED related features do not combine question type features, taking out all three POS/NER/DEP features decreases F1 by 30%. Without TED related features (both EDIT and ALIGN) F1 also drops more than 10%.

Figure 4 is a bar chart showing how much improvement each feature brings. While having a baseline model with 31.4% in F1, traditional features based on POS/DEP/NER and question types brings a 10% increase with a simple sequence tagging model (second bar labeled "CHUNKING" in the figure). Furthermore, adding TED based features to the model boosted F1 by another 10%.

5 Conclusion

Answer extraction is an essential task for any text-based question-answering system to perform. In this paper, we have cast answer extraction as a sequence tagging problem by deploying a fast and compact CRF model with simple features that capture many of the intuitions in prior "deep pipeline" approaches. We introduced novel features based on TED that boosted F1 score by 10% compared with the use of more standard features. Besides answer extraction, our modified design of the TED model is the state of the art in the task of ranking QA pairs. Finally, to improve the community's ability to evaluate QA components without requiring increasingly impractical end-to-end implementations, we have proposed answer extraction as a subtask worth evaluating in its own right, and contributed a dataset that could become a potential standard for this purpose. We believe all these developments will contribute to the continuing improvement of QA systems in the future.

Acknowledgement We thank Vulcan Inc. for funding this work. We also thank Michael Heilman and Mengqiu Wang for helpful discussion and dataset, and the three anonymous reviewers for insightful comments.

References

Nikolaus Augsten, Denilson Barbosa, Michael Böhlen, and Themis Palpanas. 2010. TASM: Top-k Approximate Subtree Matching. In *Proceedings of the International Conference on Data Engineering (ICDE-10)*, pages 353–364, Long Beach, California, USA, March. IEEE Computer Society.

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231.

P. Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1):217–239.

P. Blunsom, K. Kocik, and J.R. Curran. 2006. Question classification with log-linear models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616. ACM.

Peter Clark, Vinay Chaudhri, Sunil Mishra, Jérôme Thoméré, Ken Barker, and Bruce Porter. 2003. Enabling domain experts to convey questions to a machine: a modified, template-based approach. In *Proceedings of the 2nd international conference on Knowledge Capture*, pages 13–19, New York, NY, USA. ACM.

Shilin Ding, Gao Cong, Chin yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. In *In Proceedings of ACL-08: HLT*.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A.A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J. Prager, et al. 2010. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79.

K. Hacioglu and W. Ward. 2003. Question classification with support vector machines and error correcting codes. In *Proceedings of NAACL 2003, short papers*, pages 28–30.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL 2010*, pages 1011–1019, Los Angeles, California.

U. Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–6.

Milen Kouylekov and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *PASCAL Challenges on RTE*, pages 17–20.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL 2002*, pages 1–7.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. 2002. Is it the right answer?: exploiting web redundancy for answer validation. In *Proceedings of ACL 2002*, pages 425–432.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98.

D. Metzler and W.B. Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504.

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings of ACL 2007*, volume 45, page 776.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs).

A. Penas, A. Rodrigo, V. Sama, and F. Verdejo. 2008. Testing the reasoning for question answering validation. *Journal of Logic and Computation*, 18(3):459–474.

Vasin Punyakanok, Dan Roth, and Wen T. Yih. 2004. Mapping Dependencies Trees: An Application to Question Answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP 1996*, volume 1, pages 133–142.

Deepak Ravichandran, Abraham Ittycheriah, and Salim Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of NAACL 2003, short papers*, pages 85–87, Stroudsburg, PA, USA.

D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. 2004. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th international conference on World Wide Web*, pages 502–511, New York, NY, USA. ACM.

Á. Rodrigo, A. Peñas, and F. Verdejo. 2009. Overview of the answer validation exercise 2008. *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 296–313.

H. Shima, N. Lao, E. Nyberg, and T. Mitamura. 2008. Complex cross-lingual question answering as sequential classification and multi-document summarization task. In *Proceedings of NTICIR-7 Workshop, Japan*.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30, New York, June.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632, New York, NY, USA. ACM.

E. Sneiders. 2002. *Automated question answering: template-based approach*. Ph.D. thesis, KTH.

Martin M. Soubbotin. 2001. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of ACL 2010*, pages 1164–1172, Stroudsburg, PA, USA.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic, June.

D. Zhang and W.S. Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM.

K. Zhang and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, December.