What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA

Mengqiu Wang and **Noah A. Smith** and **Teruko Mitamura**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{mengqiu,nasmith,teruko}@cs.cmu.edu

Abstract

This paper presents a syntax-driven approach to question answering, specifically the answer-sentence selection problem for short-answer questions. Rather than using syntactic features to augment existing statistical classifiers (as in previous work), we build on the idea that questions and their (correct) answers relate to each other via loose but predictable syntactic transformations. We propose a probabilistic quasi-synchronous grammar, inspired by one proposed for machine translation (D. Smith and Eisner, 2006), and parameterized by mixtures of a robust non-lexical syntax/alignment model with a(n optional) lexical-semantics-driven log-linear model. Our model learns soft alignments as a hidden variable in discriminative training. Experimental results using the TREC dataset are shown to significantly outperform strong state-of-the-art baselines.

1 Introduction and Motivation

Open-domain question answering (QA) is a widely-studied and fast-growing research problem. State-of-the-art QA systems are extremely complex. They usually take the form of a pipeline architecture, chaining together modules that perform tasks such as answer type analysis (identifying whether the correct answer will be a person, location, date, etc.), document retrieval, answer candidate extraction, and answer reranking. This architecture is so predominant that each task listed above has evolved into its own sub-field and is often studied and evaluated independently (Shima et al., 2006).

At a high level, the QA task boils down to only two essential steps (Echihabi and Marcu, 2003). The first step, **retrieval**, narrows down the search space from a corpus of millions of documents to a focused set of maybe a few hundred using an IR engine, where efficiency and recall are the main focus. The second step, **selection**, assesses each candidate answer string proposed by the first step, and finds the one that is most likely to be an answer to the given question. The granularity of the target answer string varies depending on the type of the question. For example, answers to factoid questions (e.g., Who, When, Where) are usually single words or short phrases, while definitional questions and other more complex question types (e.g., How, Why) look for sentences or short passages. In this work, we fix the granularity of an answer to a single sentence.

Earlier work on answer selection relies only on the surface-level text information. Two approaches are most common: surface pattern matching, and similarity measures on the question and answer, represented as bags of words. In the former, patterns for a certain answer type are either crafted manually (Soubbotin and Soubbotin, 2001) or acquired from training examples automatically (Ittycheriah et al., 2001; Ravichandran et al., 2003; Licuanan and Weischedel, 2003). In the latter, measures like cosine-similarity are applied to (usually) bag-of-words representations of the question and answer. Although many of these systems have achieved very good results in TREC-style evaluations, shallow methods using the bag-of-word representation clearly have their limitations. Examples of

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 22–32, Prague, June 2007.

cases where the bag-of-words approach fails abound in QA literature; here we borrow an example used by Echihabi and Marcu (2003). The question is "*Who is the leader of France?*", and the sentence "Henri Hadjenberg, *who is the leader of France* 's Jewish community, endorsed ..." (note tokenization), which is not the correct answer, matches all keywords in the question in exactly the same order. (The correct answer is found in "Bush later met with French President Jacques Chirac.")

This example illustrates two types of variation that need to be recognized in order to connect this question-answer pair. The first variation is the change of the word "leader" to its semantically related term "president". The second variation is the syntactic shift from "leader of France" to "French president." It is also important to recognize that "France" in the first sentence is modifying "community", and therefore "Henri Hadjenberg" is the "leader of ... community" rather than the "leader of France." These syntactic and semantic variations occur in almost every question-answer pair, and typically they cannot be easily captured using shallow representations. It is also worth noting that such syntactic and semantic variations are not unique to QA; they can be found in many other closely related NLP tasks, motivating extensive community efforts in syntactic and semantic processing.

Indeed, in this work, we imagine a generative story for QA in which the question is generated from the answer sentence through a series of syntactic and semantic transformations. The same story has been told for **machine translation** (Yamada and Knight, 2001, *inter alia*), in which a target language sentence (the desired output) has undergone semantic transformation (word to word translation) and syntactic transformation (syntax divergence across languages) to generate the source language sentence (noisy-channel model). Similar stories can also be found in **paraphrasing** (Quirk et al., 2004; Wu, 2005) and **textual entailment** (Harabagiu and Hickl, 2006; Wu, 2005).

Our story makes use of a weighted formalism known as **quasi-synchronous grammar** (hereafter, QG), originally developed by D. Smith and Eisner (2006) for machine translation. Unlike most synchronous formalisms, QG does not posit a strict isomorphism between the two trees, and it provides an elegant description for the set of local configurations. In Section 2 we situate our contribution in the context of earlier work, and we give a brief discussion of quasi-synchronous grammars in Section 3. Our version of QG, called the **Jeopardy model**, and our parameter estimation method are described in Section 4. Experimental results comparing our approach to two state-of-the-art baselines are presented in Section 5. We discuss portability to cross-lingual QA and other applied semantic processing tasks in Section 6.

2 Related Work

To model the syntactic transformation process, researchers in these fields—especially in machine translation—have developed powerful grammatical formalisms and statistical models for representing and learning these tree-to-tree relations (Wu and Wong, 1998; Eisner, 2003; Gildea, 2003; Melamed, 2004; Ding and Palmer, 2005; Quirk et al., 2005; Galley et al., 2006; Smith and Eisner, 2006, *inter alia*). We can also observe a trend in recent work in textual entailment that more emphasis is put on explicit learning of the syntactic graph mapping between the entailed and entailed-by sentences (MacCartney et al., 2006).

However, relatively fewer attempts have been made in the QA community. As pointed out by Katz and Lin (2003), most early experiments in QA that tried to bring in syntactic or semantic features showed little or no improvement, and it was often the case that performance actually degraded (Litkowski, 1999; Attardi et al., 2001). More recent attempts have tried to augment the bag-of-words representation—which, after all, is simply a real-valued feature vector—with syntactic features. The usual similarity measures can then be used on the new feature representation. For example, Punyakanok et al. (2004) used approximate tree matching and tree-edit-distance to compute a similarity score between the question and answer parse trees. Similarly, Shen et al. (2005) experimented with dependency tree kernels to compute similarity between parse trees. Cui et al. (2005) measured sentence similarity based on similarity measures between dependency paths among aligned words. They used heuristic functions similar to mutual information to assign scores to matched pairs of dependency links. Shen and Klakow (2006) extend the idea further through the use of log-linear models to learn a scoring function for relation pairs.

Echihabi and Marcu (2003) presented a noisy-channel approach in which they adapted the IBM model 4 from statistical machine translation (Brown et al., 1990; Brown et al., 1993) and applied it to QA. Similarly, Murdock and Croft (2005) adopted a simple translation model from IBM model 1 (Brown et al., 1990; Brown et al., 1993) and applied it to QA. Porting the translation model to QA is not straightforward; it involves parse-tree pruning heuristics (the first two deterministic steps in Echihabi and Marcu, 2003) and also replacing the lexical translation table with a monolingual "dictionary" which simply encodes the identity relation. This brings us to the question that drives this work: is there a statistical translation-like model that is natural and accurate for question answering? We propose Smith and Eisner's (2006) quasi-synchronous grammar (Section 3) as a general solution and the Jeopardy model (Section 4) as a specific instance.

3 Quasi-Synchronous Grammar

For a formal description of QG, we recommend Smith and Eisner (2006). We briefly review the central idea here. QG arose out of the empirical observation that translated sentences often have *some* isomorphic syntactic structure, but not usually in entirety, and the strictness of the isomorphism may vary across words or syntactic rules. The idea is that, rather than a synchronous structure over the source and target sentences, a tree over the target sentence is modeled by a source-sentence-specific grammar that is inspired by the source sentence's tree.[1] This is implemented by a "sense"—really just a subset of nodes in the *source* tree—attached to each grammar node in the *target* tree. The senses define an alignment between the trees. Because it only loosely links the two sentences' syntactic structure, QG is particularly well-suited for QA insofar as QA is like "free" translation.

A concrete example that is easy to understand is a binary quasi-synchronous context-free grammar (denoted QCFG). Let V_S be the set of constituent tokens in the source tree. QCFG rules would take the augmented form

$$\begin{array}{rcl} \langle X, \mathcal{S}_1 \rangle & \rightarrow & \langle Y, \mathcal{S}_2 \rangle \langle Z, \mathcal{S}_3 \rangle \\ \langle X, \mathcal{S}_1 \rangle & \rightarrow & w \end{array}$$

where X, Y, and Z are ordinary CFG nonterminals, each $\mathcal{S}_i \in 2^{V_S}$ (subsets of nodes in the source tree to which the nonterminals align), and w is a target-language word. QG can be made more or less "liberal" by constraining the cardinality of the $\mathcal{S}_i$ (we force all $|\mathcal{S}_i| = 1$), and by constraining the relationships *among* the $\mathcal{S}_i$ mentioned in a single rule. These are called permissible "configurations." An example of a strict configuration is that a target parent-child pair must align (respectively) to a *source* parent-child pair. Configurations are shown in Table 1.

Here, following Smith and Eisner (2006), we use a weighted, quasi-synchronous *dependency* grammar. Apart from the obvious difference in application task, there are a few important differences with their model. First, we are not interested in the alignments *per se*; we will sum them out as a hidden variable when scoring a question-answer pair. Second, our probability model includes an optional mixture component that permits arbitrary features—we experiment with a small set of WordNet lexical-semantics features (see Section 4.4). Third, we apply a more discriminative training method (conditional maximum likelihood estimation, Section 4.5).

4 The Jeopardy Model

Our model, informally speaking, aims to follow the process a player of the television game show *Jeopardy!* might follow. The player knows the answer (or at least *thinks* he knows the answer) and must quickly turn it into a question.[2] The question-answer pairs used on *Jeopardy!* are not precisely what we have in mind for the real task (the questions are not specific enough), but the syntactic transformation inspires our model. In this section we formally define

[1] Smith and Eisner also show how QG formalisms generalize synchronous grammar formalisms.

[2] A round of *Jeopardy!* involves a somewhat involved and specific "answer" presented to the competitors, and the first competitor to hit a buzzer proposes the "question" that leads to the answer. For example, an answer might be, *This Eastern European capital is famous for defenestrations.* In *Jeopardy!* the players must respond with a queston: *What is Prague?*

this probability model and present the necessary algorithms for parameter estimation.

4.1 Probabilistic Model

The Jeopardy model is a QG designed for QA. Let $\mathbf{q} = \langle q_1, ..., q_n \rangle$ be a question sentence (each q_i is a word), and let $\mathbf{a} = \langle a_1, ..., a_m \rangle$ be a candidate answer sentence. (We will use $\mathbf{w}$ to denote an abstract sequence that could be a question or an answer.) In practice, these sequences may include other information, such as POS, but for clarity we assume just words in the exposition. Let $\mathcal{A}$ be the set of candidate answers under consideration. Our aim is to choose:

$$\hat{\mathbf{a}} = \operatorname*{argmax}_{\mathbf{a} \in \mathcal{A}} p(\mathbf{a} \mid \mathbf{q}) \quad (1)$$

At a high level, we make three adjustments. The first is to apply Bayes' rule, $p(\mathbf{a} \mid \mathbf{q}) \propto p(\mathbf{q} \mid \mathbf{a}) \cdot p(\mathbf{a})$. Because $\mathcal{A}$ is known and is assumed to be generated by an external extraction system, we could use that extraction system to assign scores (and hence, probabilities $p(\mathbf{a})$) to the candidate answers. Other scores could also be used, such as reputability of the document the answer came from, grammaticality, etc. Here, aiming for simplicity, we do not aim to use such information. Hence we treat $p(\mathbf{a})$ as uniform over $\mathcal{A}$.[3]

The second adjustment adds a labeled, directed **dependency tree** to the question and the answer. The tree is produced by a state-of-the-art dependency parser (McDonald et al., 2005) trained on the *Wall Street Journal* Penn Treebank (Marcus et al., 1993). A dependency tree on a sequence $\mathbf{w} = \langle w_1, ..., w_k \rangle$ is a mapping of indices of words to indices of their syntactic parents and a label for the syntactic relation, $\tau : \{1, ..., k\} \rightarrow \{0, ..., k\} \times \mathcal{L}$. Each word w_i has a single parent, denoted $w_{\tau(i).par}$. Cycles are not permitted. w_0 is taken to be the invisible "wall" symbol at the left edge of the sentence; it has a single child ($|\{i : \tau(i) = 0\}| = 1$). The label for w_i is denoted $\tau(i).lab$.

The third adjustment involves a hidden variable X, the **alignment** between question and answer words. In our model, each question-word maps to exactly one answer-word. Let $x : \{1, ..., n\} \rightarrow \{1, ..., m\}$ be a mapping from indices of words in $\mathbf{q}$ to indices of words in $\mathbf{a}$. (It is for computational reasons that we assume $|x(i)| = 1$; in general x could range over *subsets* of $\{1, ..., m\}$.) Because we define the correspondence in this direction, note that it is possible for multple question words to map to the same answer word.

Why do we treat the alignment X as a hidden variable? In prior work, the alignment is assumed to be known given the sentences, but we aim to discover it from data. Our guide in this learning is the structure inherent in the QG: the configurations between parent-child pairs in the question and their corresponding, aligned words in the answer. The hidden variable treatment lets us avoid commitment to any one x mapping, making the method more robust to noisy parses (after all, the parser is not 100% accurate) and any wrong assumptions imposed by the model (that $|x(i)| = 1$, for example, or that syntactic transformations can explain the connection between $\mathbf{q}$ and $\mathbf{a}$ at all).[4]

Our model, then, defines

$$p(\mathbf{q}, \tau_\mathbf{q} \mid \mathbf{a}, \tau_\mathbf{a}) = \sum_x p(\mathbf{q}, \tau_\mathbf{q}, x \mid \mathbf{a}, \tau_\mathbf{a}) \quad (2)$$

where $\tau_\mathbf{q}$ and $\tau_\mathbf{a}$ are the question tree and answer tree, respectively. The stochastic process defined by our model factors cleanly into recursive steps that derive the question from the top down. The QG defines a grammar for this derivation; the grammar depends on the specific answer.

Let $\tau_\mathbf{w}^i$ refer to the subtree of $\tau_\mathbf{w}$ rooted at w_i. The model is defined by:

$$\begin{aligned} & p(\tau_\mathbf{q}^i \mid q_i, \tau_\mathbf{q}(i), x(i), \tau_\mathbf{a}) = \quad (3) \\ & \quad p_{\#kids}(|\{j : \tau_\mathbf{q}(j) = i, j < i\}| \mid q_i, \textit{left}) \\ & \quad \times p_{\#kids}(|\{j : \tau_\mathbf{q}(j) = i, j > i\}| \mid q_i, \textit{right}) \\ & \quad \times \prod_{j : \tau_\mathbf{q}(j) = i} \sum_{x(j)=0}^{m} \\ & \qquad p_{kid}(q_j, \tau_\mathbf{q}(j).lab \mid q_i, \tau_\mathbf{q}(i), x(i), x(j), \tau_\mathbf{a}) \\ & \qquad \times p(\tau_\mathbf{q}^j \mid q_j, \tau_\mathbf{q}(j), x(j), \tau_\mathbf{a}) \end{aligned}$$

[3] The main motivation for modeling $p(\mathbf{q} \mid \mathbf{a})$ is that it is easier to model *deletion* of information (such as the part of the sentence that answers the question) than *insertion*. Our QG does not model the real-world knowledge required to fill in an answer; its job is to know what answers are likely to look like, syntactically.

[4] If parsing performance is a concern, we might also treat the question and/or answer parse trees as hidden variables, though that makes training and testing more computationally expensive.

Note the recursion in the last line. While the above may be daunting, in practice it boils down only to defining the conditional distribution p_{kid}, since the number of left and right children of each node need not be modeled (the trees are assumed known)—$p_{\#kids}$ is included above for completeness, but in the model applied here we do not condition it on q_i and therefore do not need to estimate it (since the trees are fixed).

p_{kid} defines a distribution over syntactic children of q_i and their labels, given (1) the word q_i, (2) the parent of q_i, (3) the dependency relation between q_i and its parent, (4) the answer-word q_i is aligned to, (5) the answer-word the *child* being predicted is aligned to, and (6) the remainder of the answer tree.

4.2 Dynamic Programming

Given $\mathbf{q}$, the score for an answer is simply $p(\mathbf{q}, \tau_{\mathbf{q}} \mid \mathbf{a}, \tau_{\mathbf{a}})$. Computing the score requires summing over alignments and can be done efficiently by bottom-up dynamic programming. Let $S(j, \ell)$ refer to the score of $\tau_{\mathbf{q}}^j$, assuming that the parent of q_j, $\tau_{\mathbf{q}}(j).par$, is aligned to a_ℓ. The base case, for leaves of $\tau_{\mathbf{q}}$, is:

$$\begin{aligned} S(j,\ell) = & \qquad\qquad (4) \\ & p_{\#kids}(0 \mid q_j, \mathit{left}) \times p_{\#kids}(0 \mid q_j, \mathit{right}) \\ & \times \sum_{k=0}^{m} p_{kid}(q_j, \tau_{\mathbf{q}}(j).lab \mid q_{\tau_{\mathbf{q}}(j)}, \ell, k, \tau_{\mathbf{a}}) \end{aligned}$$

Note that k ranges over indices of answer-words to be aligned to q_j. The recursive case is

$$\begin{aligned} S(i,\ell) = & \qquad\qquad (5) \\ & p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j < i\}| \mid q_j, \mathit{left}) \\ & \times p_{\#kids}(|\{j : \tau_{\mathbf{q}}(j) = i, j > i\}| \mid q_j, \mathit{right}) \\ & \times \sum_{k=0}^{m} p_{kid}(q_i, \tau_{\mathbf{q}}(i).lab \mid q_{\tau_{\mathbf{q}}(i)}, \ell, k, \tau_{\mathbf{a}}) \\ & \quad \times \prod_{j:\tau_{\mathbf{q}}(j)=i} S(j,k) \end{aligned}$$

Solving these equations bottom-up can be done in $O(nm^2)$ time and $O(nm)$ space; in practice this is very efficient. In our experiments, computing the value of a question-answer pair took two seconds on average.[5] We turn next to the details of p_{kid}, the core of the model.

4.3 Base Model

Our base model factors p_{kid} into three conditional multinomial distributions.

$$\begin{aligned} & p_{kid}^{base}(q_i, \tau_{\mathbf{q}}(i).lab \mid q_{\tau_{\mathbf{q}}(i)}, \ell, k, \tau_{\mathbf{a}}) = \\ & \quad p(q_i.pos \mid a_k.pos) \times p(q_i.ne \mid a_k.ne) \\ & \quad \times p(\tau_{\mathbf{q}}(i).lab \mid \mathit{config}(\tau_{\mathbf{q}}, \tau_{\mathbf{a}}, i)) \qquad (6) \end{aligned}$$

where $q_i.pos$ is question-word i's POS label and $q_i.ne$ is its named-entity label. *config* maps question-word i, its parent, and their alignees to a QG configuration as described in Table 1; note that some configurations are extended with additional tree information. The base model does not directly predict the specific words in the question—only their parts-of-speech, named-entity labels, and dependency relation labels. This model is very similar to Smith and Eisner (2006).

Because we are interested in augmenting the QG with additional lexical-semantic knowledge, we also estimate p_{kid} by **mixing** the base model with a model that exploits WordNet (Miller et al., 1990) lexical-semantic relations. The mixture is given by:

$$p_{kid}(\bullet \mid \bullet) = \alpha p_{kid}^{base}(\bullet \mid \bullet) + (1-\alpha) p_{kid}^{ls}(\bullet \mid \bullet) \qquad (7)$$

4.4 Lexical-Semantics Log-Linear Model

The lexical-semantics model p_{kid}^{ls} is defined by predicting a (nonempty) subset of the thirteen classes for the question-side word given the identity of its aligned answer-side word. These classes include WordNet relations: identical-word, synonym, antonym (also extended and indirect antonym), hypernym, hyponym, derived form, morphological variation (e.g., plural form), verb group, entailment, entailed-by, see-also, and causal relation. In addition, to capture the special importance of Wh-words in questions, we add a special semantic relation called "q-word" between any word and any Wh-word. This is done through a log-linear model with one feature per relation. Multiple relations may fire, motivating the log-linear model, which permits "overlapping" features, and, therefore prediction of

[5] Experiments were run on a 64-bit machine with 2× 2.2GHz dual-core CPUs and 4GB of memory.

any of the possible $2^{13} - 1$ nonempty subsets. It is important to note that this model assigns zero probability to alignment of an answer-word with any question-word that is *not* directly related to it through *any* relation. Such words may be linked in the mixture model, however, via p_{kid}^{base}.[6]

(It is worth pointing out that log-linear models provide great flexibility in defining new features. It is straightforward to extend the feature set to include more domain-specific knowledge or other kinds of morphological, syntactic, or semantic information. Indeed, we explored some additional syntactic features, fleshing out the configurations in Table 1 in more detail, but did not see any interesting improvements.)

parent-child	Question parent-child pair align respectively to answer parent-child pair. Augmented with the q.-side dependency label.
child-parent	Question parent-child pair align respectively to answer child-parent pair. Augmented with the q.-side dependency label.
grandparent-child	Question parent-child pair align respectively to answer grandparent-child pair. Augmented with the q.-side dependency label.
same node	Question parent-child pair align to the same answer-word.
siblings	Question parent-child pair align to siblings in the answer. Augmented with the tree-distance between the a.-side siblings.
c-command	The parent of one answer-side word is an ancestor of the other answer-side word.
other	A catch-all for all other types of configurations, which are permitted.

Table 1: Syntactic alignment configurations are partitioned into these sets for prediction under the Jeopardy model.

4.5 Parameter Estimation

The parameters to be estimated for the Jeopardy model boil down to the conditional multinomial distributions in p_{kid}^{base}, the log-linear weights inside of p_{kid}^{ls}, and the mixture coefficient α.[7] Standard applications of log-linear models apply conditional maximum likelihood estimation, which for our case involves using an empirical distribution $\tilde{p}$ over question-answer pairs (and their trees) to optimize as follows:

$$\max_{\theta} \sum_{\mathbf{q},\tau_\mathbf{q},\mathbf{a},\tau_\mathbf{a}} \tilde{p}(\mathbf{q},\tau_\mathbf{q},\mathbf{a},\tau_\mathbf{a}) \log \underbrace{p_\theta(\mathbf{q},\tau_\mathbf{q} \mid \mathbf{a},\tau_\mathbf{a})}_{\sum_x p_\theta(\mathbf{q},\tau_\mathbf{q},x \mid \mathbf{a},\tau_\mathbf{a})} \quad (8)$$

Note the hidden variable x being summed out; that makes the optimization problem non-convex. This sort of problem can be solved in principle by conditional variants of the Expectation-Maximization algorithm (Baum et al., 1970; Dempster et al., 1977; Meng and Rubin, 1993; Jebara and Pentland, 1999). We use a quasi-Newton method known as L-BFGS (Liu and Nocedal, 1989) that makes use of the gradient of the above function (straightforward to compute, but omitted for space).

5 Experiments

To evaluate our model, we conducted experiments using Text REtrieval Conference (TREC) 8–13 QA dataset.[8]

5.1 Experimental Setup

The TREC dataset contains questions and answer patterns, as well as a pool of documents returned by participating teams. Our task is the same as Punyakanok et al. (2004) and Cui et al. (2005), where we search for single-sentence answers to factoid questions. We follow a similar setup to Shen and Klakow (2006) by automatically selecting answer candidate sentences and then comparing against a human-judged gold standard.

We used the questions in TREC 8–12 for training and set aside TREC 13 questions for development (84 questions) and testing (100 questions). To generate the candidate answer set for *development* and *testing*, we automatically selected sentences from each question's document pool that contains one or more non-stopwords from the question. For generating the training candidate set, in addtion to the sentences that contain non-stopwords from the question, we also added sentences that contain correct

[6]It is to preserve that robustness property that the models are *mixed*, and not combined some other way.

[7]In our experiments, all log-linear weights are initialized to be 1; all multinomial distributions are initialized as uniform distributions; α is initialized to be 0.1.

[8]We thank the organizers and NIST for making the dataset publicly available.

answer pattern. Manual judgement was produced for the entire TREC 13 set, and also for the first 100 questions from the training set TREC 8–12.[9] On average, each question in the development set has 3.1 positive and 17.1 negative answers. There are 3.6 positive and 20.0 negative answers per question in the test set.

We tokenized sentences using the standard treebank tokenization script, and then we performed part-of-speech tagging using MXPOST tagger (Ratnaparkhi, 1996). The resulting POS-tagged sentences were then parsed using MSTParser (McDonald et al., 2005), trained on the entire Penn Treebank to produce labeled dependency parse trees (we used a coarse dependency label set that includes twelve label types). We used BBN Identifinder (Bikel et al., 1999) for named-entity tagging.

As answers in our task are considered to be single sentences, our evaluation differs slightly from TREC, where an answer string (a word or phrase like *1977* or *George Bush*) has to be accompanied by a supporting document ID. As discussed by Punyakanok et al. (2004), the single-sentence assumption does not simplify the task, since the hardest part of answer finding is to locate the correct sentence. From an end-user's point of view, presenting the sentence that contains the answer is often more informative and evidential. Furthermore, although the judgement data in our case are more labor-intensive to obtain, we believe our evaluation method is a better indicator than the TREC evaluation for the quality of an answer selection algorithm.

To illustrate the point, consider the example question, "*When did James Dean die?*" The correct answer as appeared in the sentence "*In 1955, actor James Dean was killed in a two-car collision near Cholame, Calif.*" is *1955*. But from the same document, there is another sentence which also contains *1955*: "*In 1955, the studio asked him to become a technical adviser on Elia Kazan's 'East of Eden,' starring James Dean.*" If a system missed the first sentence but happened to have extracted *1955* from the second one, the TREC evaluation grants it a "correct and well-supported" point, since the document ID matches the correct document ID—even though the latter answer does not entail the true answer. Our evaluation does not suffer from this problem.

We report two standard evaluation measures commonly used in IR and QA research: mean average precision (MAP) and mean reciprocal rank (MRR). All results are produced using the standard `trec_eval` program.

5.2 Baseline Systems

We implemented two state-of-the-art answer-finding algorithms (Cui et al., 2005; Punyakanok et al., 2004) as strong baselines for comparison. Cui et al. (2005) is the answer-finding algorithm behind one of the best performing systems in TREC evaluations. It uses a mutual information-inspired score computed over dependency trees and a single alignment between them. We found the method to be brittle, often not finding a score for a testing instance because alignment was not possible. We extended the original algorithm, allowing fuzzy word alignments through WordNet expansion; both results are reported.

The second baseline is the approximate tree-matching work by Punyakanok et al. (2004). Their algorithm measures the similarity between $\tau_{\mathbf{q}}$ and $\tau_{\mathbf{a}}$ by computing tree edit distance. Our replication is close to the algorithm they describe, with one subtle difference. Punyakanok et al. used answer-typing in computing edit distance; this is not available in our dataset (and our method does not explicitly carry out answer-typing). Their heuristics for reformulating questions into statements were not replicated. We did, however, apply WordNet type-checking and approximate, penalized lexical matching. Both results are reported.

[9] More human-judged data are desirable, though we will address training from noisy, *automatically* judged data in Section 5.4. It is important to note that human judgement of answer sentence correctness was carried out prior to any experiments, and therefore is unbiased. The total number of questions in TREC 13 is 230. We exclude from the TREC 13 set questions that either have no correct answer candidates (27 questions), or no incorrect answer candidates (19 questions). Any algorithm will get the same performance on these questions, and therefore obscures the evaluation results. 6 such questions were also excluded from the 100 manually-judged training questions, resulting in 94 questions for training. For computational reasons (the cost of parsing), we also eliminated answer candidate sentences that are longer than 40 words from the training and evaluation set. After these data preparation steps, we have 348 positive Q-A pairs for training, 1,415 Q-A pairs in the development set, and 1,703 Q-A pairs in the test set.

training dataset	model	development set MAP	development set MRR	test set MAP	test set MRR
100 manually-judged	TreeMatch	0.4074	0.4458	0.3814	0.4462
	+WN	0.4328	0.4961	0.4189	0.4939
	Cui et al.	0.4715	0.6059	0.4350	0.5569
	+WN	0.5311	0.6162	0.4271	0.5259
	Jeopardy (base only)	0.5189	0.5788	0.4828	0.5571
	Jeopardy	**0.6812**	**0.7636**	**0.6029**	**0.6852**
+2,293 noisy	Cui et al.	0.2165	0.3690	0.2833	0.4248
	+WN	0.4333	0.5363	0.3811	0.4964
	Jeopardy (base only)	0.5174	0.5570	0.4922	0.5732
	Jeopardy	0.6683	0.7443	0.5655	0.6687

Table 2: Results on development and test sets. TreeMatch is our implementation of Punyakanok et al. (2004); +WN modifies their edit distance function using WordNet. We also report our implementation of Cui et al. (2005), along with our WordNet expansion (+WN). The Jeopardy base model and mixture with the lexical-semantics log-linear model perform best; both are trained using conditional maximum likelihood estimation. The top part of the table shows performance using 100 manually-annotated question examples (questions 1–100 in TREC 8–12), and the bottom part adds noisily, automatically annotated questions 101–2,393. Boldface marks the best score in a column and any scores in that column not significantly worse under a a two-tailed paired t-test ($p < 0.03$).

5.3 Results

Evaluation results on the development and test sets of our model in comparison with the baseline algorithms are shown in Table 2. Both our model and the model in Cui et al. (2005) are trained on the manually-judged training set (questions 1-100 from TREC 8–12). The approximate tree matching algorithm in Punyakanok et al. (2004) uses fixed edit distance functions and therefore does not require training. From the table we can see that our model significantly outperforms the two baseline algorithms—even when they are given the benefit of WordNet—on both development and test set, and on both MRR and MAP.

5.4 Experiments with Noisy Training Data

Although manual annotation of the remaining 2,293 training sentences' answers in TREC 8–12 was too labor-intensive, we did experiment with a simple, noisy automatic labeling technique. Any answer that had at least three non-stop word types seen in the question *and* contains the answer pattern defined in the dataset was labeled as "correct" and used in training. The bottom part of Table 2 shows the results. Adding the noisy data hurts all methods, but the Jeopardy model maintains its lead and consistently suffers less damage than Cui et al. (2005). (The TreeMatch method of Punyakanok et al. (2004) does not use training examples.)

5.5 Summing vs. Maximizing

Unlike most previous work, our model does not try to find a single correspondence between words in the question and words in the answer, during training or during testing. An alternative method might choose the *best* (most probable) alignment, rather than the sum of all alignment scores. This involves a slight change to Equation 3, replacing the summation with a maximization. The change could be made during training, during testing, or both. Table 3 shows that summing is preferable, especially during training.

6 Discussion

The key experimental result of this work is that loose syntactic transformations are an effective way to carry out statistical question answering.

One unique advantage of our model is the mixture of a factored, multinomial-based base model and a potentially very rich log-linear model. The base model gives our model robustness, and the log-

training	decoding	test set MAP	MRR
Σ	Σ	**0.6029**	**0.6852**
Σ	$\max$	0.5822	0.6489
$\max$	Σ	0.5559	0.6250
$\max$	$\max$	0.5571	0.6365

Table 3: Experimental results on comparing summing over alignments (Σ) with maximizing ($\max$) over alignments on the test set. Boldface marks the best score in a column and any scores in that column not significantly worse under a a two-tailed paired t-test ($p < 0.03$).

linear model allows us to throw in task- or domain-specific features. Using a mixture gives the advantage of smoothing (in the base model) without having to normalize the log-linear model by summing over large sets. This powerful combination leads us to believe that our model can be easily ported to other semantic processing tasks where modeling syntactic and semantic transformations is the key, such as textual entailment, paraphrasing, and cross-lingual QA.

The traditional approach to cross-lingual QA is that translation is either a pre-processing or post-processing step done independently from the main QA task. Notice that the QG formalism that we have employed in this work was originally proposed for machine translation. We might envision transformations that are performed together to form questions from answers (or vice versa) *and* to translate—a *Jeopardy!* game in which bilingual players must ask a question in a different language than that in which the answer is posed.

7 Conclusion

We described a statistical syntax-based model that softly aligns a question sentence with a candidate answer sentence and returns a score. Discriminative training and a relatively straightforward, barely-engineered feature set were used in the implementation. Our scoring model was found to greatly outperform two state-of-the-art baselines on an answer selection task using the TREC dataset.

Acknowledgments

The authors acknowledge helpful input from three anonymous reviewers, Kevin Gimpel, and David Smith. This work is supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164.

References

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, Alessandro Tommasi, Ellen M. Voorhees, and D. K. Harman. 2001. Selectively using relations to improve precision in question answering. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns whatś in a name. *Machine Learning*, 34(1-3):211–231.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, Salvador, Brazil.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43st Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44st Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system—TREC-10. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Tony Jebara and Alex Pentland. 1999. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II (NIPS)*, pages 494–500, Denver, CO, USA.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*, Gaithersburg, MD, USA.

Jinxi Xu Ana Licuanan and Ralph Weischedel. 2003. Trec2003 qa at bbn: Answering definitional questions. In *Proceedings of the 12th Text REtrieval Conference (TREC-12)*, Gaithersburg, MD, USA.

Kenneth C. Litkowski. 1999. Question-answering using semantic relation triples. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45:503–528.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, New York, NY, USA.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernado Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43st Annual Meeting of the Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

I. Dan Melamed. 2004. Algorithms for syntax-aware statistical machine translation. In *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Baltimore, MD, USA.

Xiao-Li Meng and Donald B. Rubin. 1993. Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80:267–278.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4).

Vanessa Murdock and W. Bruce Croft. 2005. A translation model for sentence retrieval. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, Vancouver, BC, USA.

Vasin Punyakanok, Dan Roth, and Wen-Tau Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, FL, USA.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*, Ann Arbor, MI, USA.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, PA, USA.

Deepak Ravichandran, Abharam Ittycheriah, and Salim Roukos. 2003. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Sydney, Australia.

Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. 2005. Exploring syntactic relation patterns for question answering. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)*, Jeju Island, Republic of Korea.

Hideki Shima, Mengqiu Wang, Frank Lin, and Teruko Mitamura. 2006. Modular approach to error analysis and evaluation for multilingual question answering. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy.

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*, New York, NY, USA.

Martin M. Soubbotin and Sergei M. Soubbotin. 2001. Patterns for potential answer expressions as clues to the right answers. In *Proceedings of the 10th Text REtrieval Conference (TREC-10)*, Gaithersburg, MD, USA.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING)*, Montreal, Canada.

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, MI, USA.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, France.