# A Joint Model for Answer Sentence Ranking and Answer Extraction

**Md Arafat Sultan**[†] **Vittorio Castelli**[‡] **Radu Florian**[‡]
[†]Institute of Cognitive Science and Department of Computer Science,
University of Colorado, Boulder, CO
[‡]IBM T.J. Watson Research Center,
Yorktown Heights, NY
arafat.sultan@colorado.edu, vittorio@us.ibm.com, raduf@us.ibm.com

## Abstract

Answer sentence ranking and answer extraction are two key challenges in question answering that have traditionally been treated in isolation, i.e., as independent tasks. In this article, we (1) explain how both tasks are related at their core by a common quantity, and (2) propose a simple and intuitive joint probabilistic model that addresses both via joint computation but task-specific application of that quantity. In our experiments with two TREC datasets, our joint model substantially outperforms state-of-the-art systems in both tasks.

## 1 Introduction

One of the original goals of AI was to build machines that can naturally interact with humans. Over time, the challenges became apparent and language processing emerged as one of AI's most puzzling areas. Nevertheless, major breakthroughs have still been made in several important tasks; with IBM's Watson (Ferrucci et al., 2010) significantly outperforming human champions in the quiz contest *Jeopardy!*, question answering (QA) is definitely one such task.

QA comes in various forms, each supporting specific kinds of user requirements. Consider a scenario where a system is given a question and a set of sentences each of which may or may not contain an answer to that question. The goal of *answer extraction* is to extract a precise answer in the form of a short span of text in one or more of those sentences. In this form, QA meets users' immediate information needs. *Answer sentence ranking*, on the other hand, is the task of assigning a rank to each sentence so that the ones that are more likely to contain an answer are ranked higher. In this form, QA is similar to information retrieval and presents greater opportunities for further exploration and learning. In this article, we propose a novel approach to jointly solving these two well-studied yet open QA problems.

Most answer sentence ranking algorithms operate under the assumption that the degree of syntactic and/or semantic similarity between questions and answer sentences is a sufficiently strong predictor of answer sentence relevance (Wang et al., 2007; Yih et al., 2013; Yu et al., 2014; Severyn and Moschitti, 2015). On the other hand, answer extraction algorithms frequently assess candidate answer phrases based primarily on their own properties relative to the question (e.g., whether the question is a *who* question and the phrase refers to a person), making inadequate or no use of sentence-level evidence (Yao et al., 2013a; Severyn and Moschitti, 2013).

Both these assumptions, however, are simplistic, and fail to capture the core requirements of the two tasks. Table 1 shows a question, and three candidate answer sentences only one of which ($S^{(1)}$) actually answers the question. Ranking models that rely solely on text similarity are highly likely to incorrectly assign similar ranks to $S^{(1)}$ and $S^{(2)}$. Such models would fail to utilize the key piece of evidence against $S^{(2)}$ that it does not contain any temporal information, necessary to answer a *when* question. Similarly, an extraction model that relies only on the features of a candidate phrase might extract the temporal expression "the year 1666" in $S^{(3)}$ as an answer despite a clear lack of sentence-level evidence.

In view of the above, we propose a joint model

Transactions of the Association for Computational Linguistics, vol. 4, pp. 113–125, 2016. Action Editor: Noah Smith.
Submission batch: 10/2015; Revision batch: 2/2016; Published 4/2016.

| | |
|---|---|
| $\boldsymbol{Q}$ | When was the Hale Bopp comet discovered? |
| $\boldsymbol{S^{(1)}}$ | The comet was first spotted by Hale and Bopp, both US astronomers, on July 22, 1995. |
| $\boldsymbol{S^{(2)}}$ | Hale-Bopp, a large comet, was observed for the first time in China. |
| $\boldsymbol{S^{(3)}}$ | The law of gravity was discovered in the year 1666 by Sir Isaac Newton. |

Table 1: A question and three candidate answer sentences.

for answer sentence ranking and answer extraction that utilizes both sentence and phrase-level evidence to solve each task. More concretely, we (1) design task-specific probabilistic models for ranking and extraction, exploiting features of candidate answer sentences and their phrases, respectively, and (2) combine the two models in a simple, intuitive step to build a joint probabilistic model for both tasks. This two-step approach facilitates construction of new joint models from any existing solutions to the two tasks.

On a publicly available TREC dataset (Wang et al., 2007), our joint model demonstrates an improvement in ranking by over 10 MAP and MRR scores over the current state of the art. It also outperforms state-of-the-art extraction systems on two TREC datasets (Wang et al., 2007; Yao et al., 2013c).

## 2 Background

In this section, we provide a formal description of the two tasks and establish terminology that we follow in later sections. The Wang et al. (2007) dataset has been the benchmark for most recent work on the two tasks as well as our own. Therefore, we situate our description in the specific context of this dataset. We also discuss related prior work.

### 2.1 Answer Sentence Ranking

Given a question $Q$ and a set of candidate answer sentences $\{S^{(1)}, ..., S^{(N)}\}$, the goal in answer sentence ranking is to assign each $S^{(i)}$ an integer $rank_Q(S^{(i)})$ so that for any pair $(i, j)$, $rank_Q(S^{(i)}) < rank_Q(S^{(j)})$ iff $S^{(i)}$ is more likely to contain an answer to $Q$ than $S^{(j)}$.[1] Thus a smaller numeric value represents a higher rank. For example, in Table 1, $rank_Q(S^{(1)}) < rank_Q(S^{(3)})$. Tied sentences may receive adjacent ranks in any order.

In the Wang et al. (2007) dataset, each candidate answer sentence $S^{(i)}$ to a question $Q$ comes with a human-assigned 0/1 label (1: $S^{(i)}$ contains an answer to $Q$, 0: it does not). A supervised ranking model must learn to rank test answer sentences from such binary annotations in the training data.

Existing models accomplish this by learning to assign a relevance score to each $(Q, S^{(i)})$ pair; these scores then can be used to rank the sentences. QA rankers predominantly operate under the hypothesis that this relevance score is a function of the syntactic and/or semantic similarities between $Q$ and $S^{(i)}$. Wang et al. (2007), for example, learn the probability of generating $Q$ from $S^{(i)}$ using syntactic transformations under a quasi-synchronous grammar formalism. The tree edit models of Heilman and Smith (2010) and Yao et al. (2013a) compute minimal tree edit sequences to align $S^{(i)}$ to $Q$, and use logistic regression to map features of edit sequences to a relevance score. Wang and Manning (2010) employ structured prediction to compute probabilities for tree edit sequences. Yao et al. (2013b) align related phrases in $Q$ and each $S^{(i)}$ using a semi-Markov CRF model and rank candidates based on their decoding scores. Yih et al. (2013) use an array of lexical semantic similarity resources, from which they derive features for a binary classifier. Convolutional neural network models proposed by Yu et al. (2014) and Severyn and Moschitti (2015) compute distributional semantic vectors of $Q$ and $S^{(i)}$ to assess their semantic similarity.

In a contrasting approach, Severyn and Moschitti (2013) connect the question focus word in $Q$ with potential answer phrases in $S^{(i)}$ using a shallow syntactic tree representation. Importantly, unlike most rankers, their model utilizes key information in individual $S^{(i)}$ phrases which encodes the degree of type-compatibility between $Q$ and $S^{(i)}$. But it fails to robustly align concepts in $Q$ and $S^{(i)}$ due to a simplistic lemma-match policy.

Our joint model factors in both semantic similarity and question-answer type-compatibility features for ranking. Moreover, our semantic similarity features (described in Section 4) are informed by recent

[1] The $rank$ function makes sense only in the context of a set of sentences; for brevity, we suppress the second parameter in the full form of the function: $rank_Q(S^{(i)}, \{S^{(1)}, ..., S^{(N)}\})$.

advances in the area of short text similarity identification (Agirre et al., 2014; Agirre et al., 2015).

## 2.2 Answer Extraction

Given a question $Q$ and a set of candidate answer sentences $\{S^{(1)}, ..., S^{(N)}\}$, the goal in answer extraction is to extract from the latter a short chunk $C$ of text (a word or a sequence of contiguous words) which is a precise answer to $Q$. In Table 1, "July 22, 1995" and "1995" in $S^{(1)}$ are two such answers.

Each positive $(Q, S^{(i)})$ pair in the Wang et al. (2007) dataset is annotated by Yao et al. (2013a) with a gold answer chunk $C_g^{(i)}$ in $S^{(i)}$. Associated with each $Q$ is also a regexp pattern $P$ that specifies one or more gold answer chunks for $Q$. Being a regexp pattern, $P$ can accommodate variants of a gold answer chunk as well as multiple gold chunks. For instance, the pattern "1995" for the example in Table 1 matches both "July 22, 1995" and "1995". An extraction algorithm extracts an answer chunk $C$, which is matched against $P$ during evaluation.

Extraction of $C$ is a multistep process. Existing solutions adopt a generic framework, which we outline in Algorithm 1. In each $S^{(i)}$, candidate answer chunks $\boldsymbol{C}^{(i)}$ are first identified and evaluated according to some criteria (steps 1–4). The best chunk $C_*^{(i)}$ in $S^{(i)}$ is then identified (step 5). From these "locally best" chunks, groups of equivalent chunks are formed (step 6), where some predefined criteria for chunk equivalence are used (e.g., non-zero word overlap). The quality of each group is computed as an aggregate over the qualities of its member chunks (steps 7–8), and finally a representative chunk from the best group is extracted as $C$ (steps 9–10).

There are, however, details that need to be filled in within this generic framework, specifically in steps 2, 4, 6 and 10 of the algorithm. Solutions differ in these specifics. Here we discuss two state-of-the-art systems (Yao et al., 2013a; Severyn and Moschitti, 2013), which are the only systems that have been evaluated on the Wang et al. (2007) regexp patterns.

Yao et al. (2013a) use a conditional random field (CRF) to simultaneously identify chunks (step 2) and compute their $\phi$ values (step 4). Their chunking features include the POS, DEP and NER tags of words. Additional features are employed for chunk quality estimation, e.g., the question type and focus, properties of the edit operation associated with the word according to their tree edit model (see Section 2.1), and so on. Severyn and Moschitti (2013) employ a two-step process. First, they extract all NP chunks for step 2, as other types of chunks rarely contain answers to TREC-style factoid questions. A kernel-based binary classifier is then trained to compute a score for each chunk (step 4). Relational links established between expected answer types and compatible chunk entity types (e.g., HUM ↔ PERSON, DATE ↔ DATE/TIME/NUMBER) provide the information necessary for classification.

```
Algorithm 1: Answer Extraction Framework
Input:
    1. $Q$: a question sentence.
    2. $\{S^{(1)}, ..., S^{(N)}\}$: candidate answer sentences.
Output: $C$: a short and precise answer to $Q$.
1  for $i \in \{1, ..., N\}$ do
2      $\boldsymbol{C}^{(i)} \leftarrow$ candidate chunks in $S^{(i)}$
3      for $c \in \boldsymbol{C}^{(i)}$ do
4          $\phi(c) \leftarrow$ quality of $c$ as an answer to $Q$
5      $C_*^{(i)} \leftarrow \arg\max_{c \in \boldsymbol{C}^{(i)}}(\phi(c))$
6  $\{\boldsymbol{G}_C^{(1)}, ..., \boldsymbol{G}_C^{(M)}\} \leftarrow$ groups of chunks in
   $\{C_*^{(1)}, ..., C_*^{(N)}\}$ s.t. chunks in each $\boldsymbol{G}_C^{(i)}$ are
   semantically equivalent under some criteria
7  for $g \in \{\boldsymbol{G}_C^{(1)}, ..., \boldsymbol{G}_C^{(M)}\}$ do
8      $\phi(g) \leftarrow \sum_{c \in g} \phi(c)$
9  $\boldsymbol{G}_C^{(*)} \leftarrow \arg\max_{g \in \{\boldsymbol{G}_C^{(1)}, ..., \boldsymbol{G}_C^{(M)}\}}(\phi(g))$
10 $C \leftarrow$ a member of $\boldsymbol{G}_C^{(*)}$
```

For step 6, both systems rely on a simple word overlap strategy: chunks with common content words are grouped together. Neither article discusses the specifics of step 10.

We adhere to this generic framework with our own models and features; but importantly, through the use of sentence-level evidence in step 4, our joint model demonstrates a substantial improvement in accuracy.

## 2.3 Coupled Ranking and Extraction

Yao et al. (2013c) present a ranker that utilizes token-level extraction features. The question sentence is augmented with such features to formulate a search query, which is fed as input to a search engine for ranked retrieval from a pool of candidate answer sentences. They experimentally show that downstream extraction from top retrievals in this list is more accurate than if the query is not expanded with the extraction features.

We take a different approach where numeric predictions from separate ranking and extraction modules are combined to jointly perform *both* tasks (Section 3). Yao et al. build on an existing ranker that supports query expansion and token-level characterization of candidate answer sentences. We assume no such system features, facilitating coupling of arbitrary models including new experimental ones. For extraction, Yao et al. simply rely on better upstream ranking, whereas our joint model provides a precise mathematical formulation of answer chunk quality as a function of both chunk and sentence relevance to the question. We observe a large increase in end-to-end extraction accuracy over the Yao et al. model in our experiments.

## 3 Approach

We first train separate probabilistic models for answer sentence ranking and answer extraction, for each of which we take an approach similar to that of existing models. Probabilities learned by the two task-specific models are then combined to construct our joint model. This section discusses the details of this two-step process.

### 3.1 Answer Sentence Ranking

Let the following logistic function represent the probability that a candidate answer sentence $S^{(i)}$ contains an answer to a question $Q$:

$$P(S^{(i)}|Q) = \frac{1}{1 + e^{-\boldsymbol{\theta}_r^T \boldsymbol{f}_r(Q, S^{(i)})}} \tag{1}$$

where $\boldsymbol{f}_r(Q, S^{(i)})$ is a set of features each of which is a unique measure of semantic similarity between $Q$ and $S^{(i)}$, and $\boldsymbol{\theta}_r$ is the weight vector learned during model training. We describe our feature set for ranking in Section 4.

Given $P(S^{(i)}|Q)$ values for $i \in \{1, ..., N\}$, ranking is straightforward: $rank_Q(S^{(i)}) < rank_Q(S^{(j)})$ if $P(S^{(i)}|Q) > P(S^{(j)}|Q)$. Note that a smaller numeric value represents a higher rank.

### 3.2 Answer Extraction

We follow the framework in Algorithm 1 for answer extraction. Below we describe our implementation of the generic steps:

1. Step 2: We adopt the strategy of (Severyn and Moschitti, 2013) of extracting only the NP chunks, for which we use a regexp chunker.
2. Step 4: The quality $\phi(c)$ of a candidate chunk $c$ in $S^{(i)}$ is given by the following logistic function:

   $$\phi(c) = P(c|Q, S^{(i)}) = \frac{1}{1 + e^{-\boldsymbol{\theta}_e^T \boldsymbol{f}_e(Q, S^{(i)}, c)}} \tag{2}$$

   where $\boldsymbol{f}_e(Q, S^{(i)}, c)$ is the feature set for chunk $c$ relative to $Q$, and $\boldsymbol{\theta}_e$ is the weight vector learned during model training. Our feature set for extraction is described in Section 5.
3. Step 6: Given an existing set of (possibly empty) chunk groups $\{\boldsymbol{G}_C^{(1)}, ..., \boldsymbol{G}_C^{(M)}\}$, a new chunk $c$ is added to group $\boldsymbol{G}_C^{(i)}$, if (1) all content words in $c$ are in at least one member of $\boldsymbol{G}_C^{(i)}$, or (2) there exists a member of $\boldsymbol{G}_C^{(i)}$ all of whose content words are in $c$. If no such group is found, a new group $\boldsymbol{G}_C^{(M+1)}$ is created with $c$ as its only member.
4. Step 10: We extract the longest chunk in $\boldsymbol{G}_C^{(*)}$ as the best answer $C$.

Additionally, we retain only the top $t$ of all the answer candidates extracted in step 5 to prevent propagation of noisy chunks to later steps. The value of $t$ is set using the Wang et al. (2007) DEV set.

### 3.3 Joint Ranking and Extraction

The primary goal of the joint model is to facilitate the application of both chunk-level and sentence-level features to ranking as well as extraction. To that end, it first computes the joint probability that (1) $S^{(i)}$ contains an answer to $Q$, and (2) $c \in \boldsymbol{C}^{(i)}$ is a correct answer chunk:

$$P(S^{(i)}, c|Q) = P(S^{(i)}|Q) \times P(c|Q, S^{(i)}) \tag{3}$$

where the two terms on the right hand side are given by Equations (1) and (2), respectively. Both ranking and extraction are then driven by task-appropriate application of this common quantity.

Given Equation (3), the condition for ranking is redefined as follows: $rank_Q(S^{(i)}) < rank_Q(S^{(j)})$ if $\max_{c \in \boldsymbol{C}^{(i)}} P(S^{(i)}, c|Q) > \max_{c \in \boldsymbol{C}^{(j)}} P(S^{(j)}, c|Q)$. This new condition rewards an $S^{(i)}$ that not only is highly semantically similar to $Q$, but also contains a chunk $c$ which is a likely answer to $Q$.

For extraction, the joint probability in Equation (3) replaces the conditional in Equation (2) for step 4 of Algorithm 1: $\phi(c) = P(S^{(i)}, c|Q)$. Again, this new definition of $\phi(c)$ rewards a chunk $c$ that is (1) type-compatible with $Q$, and (2) well-supported by the content of the containing sentence $S^{(i)}$.

Equation (3) assigns equal weight to the ranking and the extraction model. To learn these weights from data, we implement a variation of the joint model that employs a second-level regressor:

$$P(S^{(i)}, c|Q) = \frac{1}{1 + e^{-\boldsymbol{\theta}_2^T \boldsymbol{f}_2(Q, S^{(i)}, c)}} \tag{4}$$

where the feature vector $\boldsymbol{f}_2$ consists of the two probabilities in Equations (1) and (2), and $\boldsymbol{\theta}_2$ is the weight vector. While $P(S^{(i)}, c|Q)$ is computed using a different formula in this model, the methods for ranking and extraction based on it remains the same as above.

From here on, we will refer to the models in Sections 3.1 and 3.2 as our standalone ranking and extraction models, respectively, and the models in this section as the joint probabilistic model (Equation (3)) and the stacked (regression) model (Equation (4)).

### 3.4 Learning

The standalone ranking model is trained using the $0/1$ labels assigned to $(Q, S^{(i)})$ pairs in the Wang et al. (2007) dataset. For standalone extraction, we use for training the gold chunk annotations $C_g^{(i)}$ associated with $(Q, S^{(i)})$ pairs: a candidate NP chunk in $S^{(i)}$ is considered a positive example for $(Q, S^{(i)})$ iff it contains $C_g^{(i)}$ and $S^{(i)}$ is an actual answer sentence. For both ranking and extraction, the corresponding weight vector $\boldsymbol{\theta}$ is learned by minimizing the following $L_2$-regularized loss function:

$$J(\boldsymbol{\theta}) = -\frac{1}{T} \sum_{i=1}^{T} \Big[ y^{(i)} \log(P^{(i)}) + (1 - y^{(i)}) \log(1 - P^{(i)}) \Big] + \lambda \|\boldsymbol{\theta}\|_2$$

where $T$ is the number of training examples, $y^{(i)}$ is the gold label for example $i$ and $P^{(i)}$ is the model-predicted probability of example $i$ being a positive example (given by Equations (1) and (2)).

Learning of $\boldsymbol{\theta}_2$ for the stacked model works in a similar fashion, where level 1 predictions for training QA pairs (according to Equations (1) and (2)) serve as feature vectors.

## 4 Answer Sentence Ranking Features

Instead of reinventing similarity features for our QA ranker, we derive our feature set from the winning system (Sultan et al., 2015) at the SemEval 2015 Semantic Textual Similarity (STS) task (Agirre et al., 2015). STS is an annually held SemEval competition, where systems output real-valued similarity scores for input sentence pairs. Hundreds of systems have been evaluated over the past few years (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015); our chosen system was shown to outperform all other systems from all years of SemEval STS (Sultan et al., 2015).

In order to compute the degree of semantic similarity between a question $Q$ and a candidate answer sentence $S^{(i)}$, we draw features from two sources: (1) lexical alignment between $Q$ and $S^{(i)}$, and (2) vector representations of $Q$ and $S^{(i)}$, derived from their word embeddings. While the original STS system employs ridge regression, we use these features within a logistic regression model for QA ranking.

### 4.1 Alignment Features

We align related words in $Q$ and $S^{(i)}$ using a monolingual aligner originally proposed by Sultan et al. (2014). Here we give a brief description of our implementation, which employs arguably more principled methods to solve a set of subproblems. See the original article for further details.

The aligner computes for each word pair across $Q$ and $S^{(i)}$ a semantic similarity score $sim_W \in [0, 1]$ using PPDB—a large database of lexical paraphrases developed using bilingual pivoting (Ganitkevitch et al., 2013). Specifically, it allows three different levels of similarity: 1 if the two words or their lemmas are identical, a value *ppdbSim* $\in (0, 1)$ if the word pair is present in PPDB (the XXXL database)[2], and 0

[2] `http://www.cis.upenn.edu/~ccb/ppdb/`

otherwise.

It also computes the degree of similarity $sim_C$ between the two words' contexts in their respective sentences. This similarity is computed as the sum of word similarities in two different types of contexts: (1) a dependency neighborhood of size 2 (i.e. parents, grandparents, children and grandchildren), and (2) a surface-form neighborhood of size 3 (i.e. 3 words to the left and 3 words to the right). Stop words are skipped during neighbor selection. Unlike the Sultan et al. (2014) aligner, which allows a single neighbor word to be matched to multiple similar words in the other sentence, we match neighbors using a max-weighted bipartite matching algorithm, where word similarities serve as edge weights.

Every word pair across $Q$ and $S^{(i)}$ receives a final weight given by $w * sim_W + (1 - w) * sim_C$, where $w \in [0, 1]$. While Sultan et al. use a greedy best-first algorithm to align words based on these weights, we use them as edge weights in a max-weighted bipartite matching of word pairs (details follow).

We adopt the strategy of the original aligner of starting with high-precision alignments and increasing the recall in later steps. To this end, we align in the following order: (1) identical word sequences with at least one content word, (2) named entities, (3) content words, and (4) stop words. Following the original aligner, no additional context matching is performed in step 1 since a sequence itself provides contextual evidence for its tokens. For each of steps 2–4, words/entities of the corresponding type are aligned using max-weighted bipartite matching as described above (multiword named entities are considered single units in step 2); other word types and already aligned words are discarded. The values of $w$ and $ppdbSim$ are derived using a grid search on an alignment dataset (Brockett, 2007).

Given aligned words in the QA pair, our first feature computes the proportion of aligned content words in $Q$ and $S^{(i)}$, combined:

$$sim_A(Q, S^{(i)}) = \frac{n_c^a(Q) + n_c^a(S^{(i)})}{n_c(Q) + n_c(S^{(i)})}$$

where $n_c^a(\cdot)$ and $n_c(\cdot)$ represent the number of aligned content words and the total number of content words in a sentence, respectively.

$S^{(i)}$ can be arbitrarily long and still contain an answer to $Q$. In the above similarity measure, longer answer sentences are penalized due to a larger number of unaligned words. To counter this phenomenon, we add a measure of *coverage* of $Q$ by $S^{(i)}$ to the original feature set of Sultan et al. (2015):

$$cov_A(Q, S^{(i)}) = \frac{n_c^a(Q)}{n_c(Q)}$$

### 4.2 A Semantic Vector Feature

Neural word embeddings (Mikolov et al., 2013; Baroni et al., 2014; Pennington, 2014) have been highly successful as distributional word representations in the recent past. We utilize the 400-dimensional word embeddings developed by Baroni et al. (2014)[3] to construct sentence-level embeddings for $Q$ and $S^{(i)}$, which we then compare to compute a similarity score.

To construct the vector representation $\boldsymbol{V}_S$ of a given sentence $S$, we first extract the content word lemmas $\boldsymbol{C}_S = \{C_S^{(1)}, ..., C_S^{(M)}\}$ in $S$. The vectors representing these lemmas are then added to generate the sentence vector:

$$\boldsymbol{V}_S = \sum_{i=1}^{M} \boldsymbol{V}_{C_S^{(i)}}$$

Finally, a similarity measure for $Q$ and $S^{(i)}$ is derived by taking the cosine similarity between their vector representations:

$$sim_E(Q, S^{(i)}) = \frac{\boldsymbol{V}_Q \cdot \boldsymbol{V}_{S^{(i)}}}{|\boldsymbol{V}_Q||\boldsymbol{V}_{S^{(i)}}|}$$

This simple bag-of-words model was found to augment the alignment-based feature well in the evaluations reported by Sultan et al. (2015).

$sim_A$, $cov_A$ and $sim_E$ constitute our final feature set. As we show in Section 6, this small feature set outperforms the current state of the art in answer sentence ranking.

## 5 Answer Extraction Features

As mentioned in Section 3.2, we consider only NP chunks as answer candidates for extraction. Our chunk features can be categorized into two broad groups, which we describe in this section. For the following discussion, let $(Q, S^{(i)}, c)$ be our question, answer sentence, answer chunk triple.

[3] http://clic.cimec.unitn.it/composes/semantic-vectors.html

### 5.1 Question-Independent Features

These features represent properties of $c$ independent of the nature of $Q$. For example, our first two features fire if all content words in $c$ are present in $Q$ or align to words in $Q$. Such chunks rarely contain an answer, regardless of the type of $Q$.

Yao et al. (2013a) report an observation that answer chunks often appear close to aligned content words of specific types in $S^{(i)}$. To model this phenomenon, we adopt their features specifying the distance of $c$ from the nearest aligned content word $w_a$ in $S^{(i)}$ and the POS/DEP/NER tags of $w_a$. In addition, to encode the total amount of local evidence present for $c$, we employ the proportions of aligned content words in its dependency (size = 2) and surface (size = 3) contexts in $S^{(i)}$.

### 5.2 Features Containing the Question Type

These features are of the form "question-type$|x$", where $x$ can be an elementary (i.e. unit) or composite feature. The rationale is that certain features are informative primarily in the context of certain question types (e.g., a likely answer to a *when* question is a chunk containing the NER tag DATE).

**Headword Features.** We extract the headword of $c$ and use its POS/DEP/NER tags as features (appended to the question type). A headword in the subject position of $S^{(i)}$ or with PERSON as its NER tag, for example, is a likely answer to a *who* question.

**Question Focus.** The question focus word represents the entity about which the question is being asked. For example, in "*What is the largest country in the world?*", the focus word is "*country*". For question types like *what* and *which*, properties of the question focus largely determine the nature of the answer. In the above example, the focus word indicates that GPE is a likely NER tag for the answer.

We extract the question focus using a rule-based system originally designed for a different application, under the assumption that a question could span multiple sentences. The rule-based system is loosely inspired by the work of Lally et al. (2012), from which it differs radically because the questions in the *Jeopardy!* game are expressed as answers. The focus extractor first determines the question word or words, which is then used in conjunction with the parse tree to decide whether the question word itself or some other word in the sentence is the actual focus.

We pair the headword POS/DEP/NER tags with the focus word and its POS/NER tags, and add each such pair (appended to the question type) to our feature set. There are nine features here; examples include question-type|question-focus-word|headword-pos-tag and question-type|question-focus-ner-tag|headword-ner-tag.

We also employ the true/false labels of the following propositions as features (in conjunction with the question type): (1) the question focus word is in $c$, (2) the question focus POS tag is in the POS tags of $c$, and (3) the question focus NER tag is of the form x or x_DESC, and x is in the NER tags of $c$, for some x (e.g., GPE).

**Chunk Tags.** In many cases, it is not the headword of $c$ which is the answer; for example, in $Q$: "*How many states are there in the US?*" and $c$: "*50 states*", the headword of $c$ is "*states*". To extend our unit of attention from the headword to the entire chunk, we first construct vocabularies of POS and NER tags, $\boldsymbol{V}_{pos}$ and $\boldsymbol{V}_{ner}$, from training data. For each possible tag in $\boldsymbol{V}_{pos}$, we then use the presence/absence of that tag in the POS tag sequence for $c$ as a feature (in conjunction with the question type). We repeat the process for $\boldsymbol{V}_{ner}$. For the above $c$, for instance, an informative feature which is likely to fire is: "question-type=*how-many*|the NER tags of $c$ include CARDINAL".

**Partial Alignment.** For some question types, part of a correct answer chunk is often aligned to a question word (e.g., $Q$: "*How many players are on the field during a soccer game?*", $c$: "*22 players*"). To inform our model of such occurrences, we employ two features—true/false labels of the following propositions: (1) $c$ is partially aligned, (2) $c$ is not aligned at all (each in conjunction with the question type).

## 6 Experiments

### 6.1 Data

The Wang et al. (2007) corpus is created from Text REtrieval Conference (TREC) 8–13 QA data. It consists of a set of factoid questions, and for each question, a set of candidate answer sentences. Each answer candidate is automatically drawn from a larger document based on two selection criteria: (1) a non-zero content word overlap with the question, or (2)

| Dataset | # Questions | # QA Pairs | % Positive |
|---|---|---|---|
| TRAIN-ALL | 1,229 | 53,417 | 12.0 |
| TRAIN | 94 | 4,718 | 7.4 |
| DEV | 82 | 1,148 | 19.3 |
| TEST | 100 | 1,517 | 18.7 |

Table 2: Summary of the Wang et al. (2007) corpus.

a match with the gold regexp answer pattern for the question (training only).

TRAIN pairs are drawn from TREC 8–12; DEV and TEST pairs are drawn from TREC 13. Details of the TRAIN/DEV/TEST split are given in Table 2. TRAIN-ALL is a large set of automatically judged (thus noisy) QA pairs: a sentence is considered a positive example if it matches the gold answer pattern for the corresponding question. TRAIN is a much smaller subset of TRAIN-ALL, containing pairs that are manually corrected for errors. Manual judgment is produced for DEV and TEST pairs, too.

For answer extraction, Yao et al. (2013a) add to each QA pair the correct answer chunk(s). The gold TREC patterns are used to first identify relevant chunks in each answer sentence. TRAIN, DEV and TEST are then manually corrected for errors.

The Wang et al. (2007) dataset also comes with POS/DEP/NER tags for each sentence. They use the MXPOST tagger (Ratnaparkhi, 1996) for POS tagging, the MSTParser (McDonald et al., 2005) to generate typed dependency trees, and the BBN Identifinder (Bikel et al., 1999) for NER tagging. Although we have access to a state-of-the-art information pipeline that produces better tags, this paper aims to study the effect of the proposed models and of our features on system performance, rather than on additional variables; therefore, to support comparison with prior work, we rely on the tags provided with the dataset for all our experiments.

## 6.2 Answer Sentence Ranking

We adopt the standard evaluation procedure and metrics for QA rankers reported in the literature.

### 6.2.1 Evaluation Metrics

Our metrics for ranking are Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Here we define both in terms of simpler metrics.

**Precision at $\boldsymbol{K}$.** Given a question $Q$ and a set of candidate answer sentences $\{S^{(1)}, ..., S^{(N)}\}$, let the output of a ranker be $[R^{(1)}, ..., R^{(N)}]$, so that each $R^{(i)} \in \{S^{(1)}, ..., S^{(N)}\}$ and the predicted rank of $R^{(i)}$ is higher than the predicted rank of $R^{(j)}$ whenever $i < j$. The ranker's precision at $K$ for $Q$ $(P_K(Q))$ is then defined as the proportion of correct answer sentences in the set $\{R^{(1)}, ..., R^{(K)}\}$.

**Average Precision.** Let $\boldsymbol{A}$ be the set of correct answer sentences for $Q$ in the above scenario. Then the average precision $(AP)$ of the ranker for $Q$ can be defined as: $AP(Q) = \frac{1}{|\boldsymbol{A}|}\sum_{i:R^{(i)}\in\boldsymbol{A}} P_i(Q)$.

**Reciprocal Rank.** In the above scenario, let $j$ be the smallest index in $\{1, ..., N\}$ such that $R^{(j)} \in \boldsymbol{A}$. Then the reciprocal rank $(RR)$ of the ranker for $Q$ is: $RR(Q) = P_j(Q) = 1/j$.

**MAP.** The MAP of a ranker over a set of questions $\boldsymbol{Q} = \{Q^{(1)}, ..., Q^{(M)}\}$ is defined as: $MAP(\boldsymbol{Q}) = \frac{1}{M}\sum_{i=1}^{M} AP(Q^{(i)})$.

**MRR.** The MRR of a ranker over a set of questions $\boldsymbol{Q} = \{Q^{(1)}, ..., Q^{(M)}\}$ is defined as: $MRR(\boldsymbol{Q}) = \frac{1}{M}\sum_{i=1}^{M} RR(Q^{(i)})$.

### 6.2.2 Setup

For QA ranking, test questions that do not have both correct and incorrect candidate answer sentences are irrelevant since any ranking is correct for such questions. Following all past QA rankers, we therefore remove such instances from DEV and TEST. Of the original 1,517 TEST pairs, 1,442 (> 95%) are retained after this exclusion.

We use the logistic regression implementation of Scikit-learn (Pedregosa et al., 2011) and use the Wang et al. (2007) DEV set to set $C$, the regularization strength parameter. The standard `trec_eval` script is used to generate all results.

### 6.2.3 Results

Table 3 shows performances of our ranking models and recent baseline systems on TEST. Our QA similarity features (i.e. the standalone ranker) outperform all baselines with both TRAIN and TRAIN-ALL, although the additional noisy examples in the latter are not found to improve results.

More importantly, we get improvements of substantially larger magnitudes using our joint models—more than 10 MAP and MRR points over the state-of-the-art system of Severyn and Moschitti (2015) with TRAIN-ALL for the joint probabilistic model.

| Model | MAP% | MRR% |
|---|---|---|
| **TRAIN** | | |
| Shnarch (2013) | 68.60 | 75.40 |
| Yih et al. (2013) | 70.92 | 77.00 |
| Yu et al. (2014) | 70.58 | 78.00 |
| Severyn & Moschitti (2015) | 73.29 | 79.62 |
| Our Standalone Model | 76.05 | 83.99 |
| Our Joint Probabilistic Model | **81.59** | **89.09** |
| Our Stacked Model | 80.77 | 86.85 |
| **TRAIN-ALL** | | |
| Yu et al. (2014) | 71.13 | 78.46 |
| Severyn & Moschitti (2015) | 74.59 | 80.78 |
| Our Standalone Model | 75.68 | 83.09 |
| Our Joint Probabilistic Model | **84.95** | **91.95** |
| Our Stacked Model | 82.56 | 90.69 |

Table 3: Answer sentence ranking results.

| Model | $P$% | $R$% | $F_1$% |
|---|---|---|---|
| **TRAIN** | | | |
| Yao et al. (2013a) | 55.2 | 53.9 | 54.5 |
| Severyn & Moschitti (2013) | 66.2 | 66.2 | 66.2 |
| Our Standalone Model | 62.9 | 62.9 | 62.9 |
| Our Joint Probabilistic Model | **69.7** | **69.7** | **69.7** |
| Our Stacked Model | 62.9 | 62.9 | 62.9 |
| **TRAIN-ALL** | | | |
| Yao et al. (2013a) | 63.6 | 62.9 | 63.3 |
| Severyn & Moschitti (2013) | 70.8 | 70.8 | 70.8 |
| Our Standalone Model | 70.8 | 70.8 | 70.8 |
| Our Joint Probabilistic Model | **76.4** | **76.4** | **76.4** |
| Our Stacked Model | 73.0 | 73.0 | 73.0 |

Table 4: Answer extraction results on the Wang et al. (2007) test set.

Unlike the standalone model, the joint models also benefit from the additional noisy examples in TRAIN-ALL. These results support the central argument of this paper that joint modeling is a better approach to answer sentence ranking.

### 6.3 Answer Extraction

We follow the procedure reported in prior work (Yao et al., 2013a; Severyn and Moschitti, 2013) to evaluate the answer chunks extracted by the system.

#### 6.3.1 Evaluation Metrics

**Precision**. Given a set of questions, the precision of an answer extraction system is the proportion of its extracted answers that are correct (i.e. match the corresponding gold regexp pattern).

**Recall**. Recall is the proportion of questions for which the system extracted a correct answer.

**$F_1$ Score.** The $F_1$ score is the harmonic mean of precision and recall. It captures the system's accuracy and coverage in a single metric.

#### 6.3.2 Setup

Following prior work, we (1) retain the 89 questions in the Wang et al. (2007) TEST set that have at least one correct answer, and (2) train only with chunks in correct answer sentences to avoid extreme bias towards *false* labels (both the standalone extraction model and stage 2 of the stacked model). As in ranking, we use Scikit-learn for logistic regression and set the regularization parameter $C$ using DEV.

#### 6.3.3 Results

Table 4 shows performances of our extraction models on the Wang et al. TEST set. The joint probabilistic model demonstrates top performance for both TRAIN and TRAIN-ALL. With TRAIN-ALL, it correctly answers 68 of the 89 test questions (5 more than the previous best model of Severyn and Moschitti (2013)). The stacked model also performs well with the larger training set. Again, these results support the central claim of the paper that answer extraction can be made better through joint modeling.

Table 5 shows performances of our standalone and joint probabilistic models (trained on TRAIN-ALL) on different TEST question types. The joint model is the better of the two across types, achieving good

| Question Type | Count | ST | JP |
|---|---|---|---|
| *what* | 37 | 51.4 | 56.8 |
| *when* | 19 | 100.0 | 100.0 |
| *where* | 11 | 100.0 | 90.9 |
| *who/whom* | 10 | 60.0 | 70.0 |
| *why* | 1 | 0.0 | 0.0 |
| *how many* | 9 | 77.8 | 100.0 |
| *how long* | 2 | 50.0 | 100.0 |

Table 5: $F_1$% of the STandalone and the Joint Probabilistic extraction model across question types.

| Question | Candidate Answer Sentence | ST | JP |
|---|---|---|---|
| How many years was Jack Welch with GE? | "Six Sigma has galvanized our company with an intensity the likes of which I have never seen in my **40 years** at GE," said John Welch, chairman of General Electric. | .517 | .113 |
| | So fervent a proselytizer is Welch that GE has spent **three years** and more than $1 billion to convert all of its divisions to the Six Sigma faith. | .714 | .090 |
| What kind of ship is the Liberty Bell 7? | Newport plans to retrieve the recovery vessel first, then go after Liberty Bell 7, the only U.S . **manned spacecraft** lost after a successful mission. | .838 | .278 |
| | "It will be a big relief" once the capsule is aboard ship, **Curt Newport** said before setting sail Thursday. | .388 | .003 |

Table 6: Scores computed by the STandalone and the Joint Probabilistic model for candidate chunks (**boldfaced**) in four (Wang et al., 2007) test sentences. Joint model scores for non-answer chunks (rows 2 and 4) are much lower.

results on all question types except *what*.

A particularly challenging subtype of *what* questions are *what be* questions, answers to which often go beyond NP chunk boundaries. A human-extracted answer to the question "*What is Muslim Brotherhood's goal?*" in the Wang et al. corpus (2007), for example, is "*advocates turning Egypt into a strict Muslim state by political means*." *What* in general is nevertheless the most difficult question type, since unlike questions like *who* or *when*, answers do not have strict categories (e.g., a fixed set of NER tags).

### 6.3.4 Qualitative Analysis

We closely examine QA pairs for which the joint probabilistic model extracts a correct answer chunk but the standalone model does not. Table 6 shows two such questions, with two candidate answer sentences for each. Candidate answer chunks are **boldfaced**.

For the first question, only the sentence in row 1 contains an answer. The standalone model assigns a higher score to the non-answer chunk in row 2, but the use of sentence-level features enables the joint model to identify the more relevant chunk in row 1. Note that the joint model score, being a product of two probabilities, is always lower than the standalone model score. However, only the relative score matters in this case, as the chunk with the highest overall score is eventually selected for extraction.

For the second question, both models compute a lower score for the non-answer chunk "Curt Newport" than the answer chunk "manned spacecraft". However, the incorrect chunk appears in several candidate answer sentences (not shown here), resulting in a high overall score for the standalone model (Algorithm 1: steps 7 and 8). The joint model assigns a much lower score to each instance of this chunk due to weak sentence-level evidence, eventually resulting in the extraction of the correct chunk.

| Model | $\boldsymbol{P}$% | $\boldsymbol{R}$% | $\boldsymbol{F_1}$% |
|---|---|---|---|
| Yao et al. (2013c) | 35.4 | 17.2 | 23.1 |
| Our Joint Probabilistic Model | **83.8** | **83.8** | **83.8** |

Table 7: Performances of two joint extraction models on the Yao et al. (2013c) test set.

### 6.3.5 A Second Extraction Dataset

Yao et al. (2013c) report an extraction dataset containing 99 test questions, derived from the MIT109 test collection (Lin and Katz, 2006) of TREC pairs. Each question in this dataset has 10 candidate answer sentences. We compare the performance of our joint probabilistic model with that of their extraction model, which extracts answers from top candidate sentences identified by their coupled ranker (Section 2.3).[4] Models are trained on their training set of 2,205 questions and 22,043 candidate QA pairs. As shown in Table 7, our model outperforms the Yao et al. model by a surprisingly large margin, correctly answering 83 of the 99 test questions.

Interestingly, our standalone model extracts six more correct answers in this dataset than the joint

[4] We compare with only their extraction model, as the larger ranking dataset is not available anymore. Precision and recall are reported at `http://cs.jhu.edu/~xuchen/packages/jacana-ir-acl2013-data-results.tar.bz2`.

| Candidate Answer Sentence | ST | JP |
|---|---|---|
| Another perk is getting to work with his son, Barry Van Dyke, who has a regular role as **Detective Steve Sloan** on "Diagnosis". | .861 | .338 |
| This is only the third time in school history the Raiders have begun a season 6-0 and the first since 1976, when **Steve Sloan**, in his second season as coach, led them to an 8-0 start and 10-2 overall record. | .494 | .010 |
| He also represented several Alabama coaches, including Ray Perkins, Bill Curry, **Steve Sloan** and Wimp Sanderson. | .334 | .007 |
| Bart Starr, Joe Namath, Ken Stabler, **Steve Sloan**, Scott Hunter and Walter Lewis are but a few of the legends on the wall of the Crimson Tide quarterbacks coach. | .334 | .009 |

Table 8: Scores computed by the STandalone and the Joint Probabilistic model for NP chunks (**boldfaced**) in four Yao et al. (2013c) test sentences for the question: *Who is the detective on 'Diagnosis Murder'?* The standalone model assigns high probabilities to non-answer chunks in the last three sentences, subsequently corrected by the joint model.

model. A close examination reveals that in all six cases, this is caused by the presence of correct answer chunks in non-answer sentences. Table 8 shows an example, where the correct answer chunk "Steve Sloan" appears in all four candidate sentences, of which only the first is actually relevant to the question. The standalone model assigns high scores to all four instances and as a result observes a high overall score for the chunk. The joint model, on the other hand, recognizes the false positives, and consequently observes a smaller overall score for the chunk. However, this desired behavior eventually results in a wrong extraction. These results have key implications for the evaluation of answer extraction systems: metrics that assess performance on individual QA pairs can enable finer-grained evaluation than what end-to-end extraction metrics offer.

## 7 Discussion

Our two-step approach to joint modeling, consisting of constructing separate models for ranking and extraction first and then coupling their predictions, offers at least two advantages. First, predictions from any given pair of ranking and extraction systems can be combined, since such systems must compute a score for a QA pair or an answer chunk in order to differentiate among candidates. Coupling of the ranking and extraction systems of Yao et al. (2013a) and Severyn and Moschitti (2013), for example, is straightforward within our framework. Second, this approach supports the use of task-appropriate training data for ranking and extraction, which can provide key advantage. For example, while answer sentence ranking systems use both correct and incorrect candidate answer sentences for model training, existing answer extraction systems discard the latter in order to maintain a (relatively) balanced class distribution (Yao et al., 2013a; Severyn and Moschitti, 2013). Through the separation of the ranking and extraction models during training, our approach naturally supports such task-specific sampling of training data.

A potentially limiting factor in our extraction model is the assumption that answers are always expressed neatly in NP chunks. While models that make no such assumption exist (e.g., the CRF model of Yao et al. (2013a)), extraction of long answers (such as the one discussed in Section 6.3.3) is still difficult in practice due to their unconstrained nature.

## 8 Conclusions and Future Work

We present a joint model for the important QA tasks of answer sentence ranking and answer extraction. By exploiting the interconnected nature of the two tasks, our model demonstrates substantial performance improvements over previous best systems for both. Additionally, our ranking model applies recent advances in the computation of short text similarity to QA, providing stronger similarity features.

An obvious direction for future work is the inclusion of new features for each task. Answer sentence ranking, for example, can benefit from phrasal alignment and long-distance context representation. Answer extraction for *what* questions can be made better using a lexical answer type feature, or world knowledge (such as "blue is a color") derived from semantic networks like WordNet. Our model also facilitates straightforward integration of features/predictions from other existing systems for both tasks, for example, the convolutional neural sentence model of Severyn and Moschitti (2015) for ranking. Finally, more sophisticated techniques are required for extraction of the final answer chunk based on individual chunk scores across QA pairs.

## Acknowledgments

We thank the reviewers for their valuable comments and suggestions. We also thank Xuchen Yao and Aliaksei Severyn for clarification of their work.

## References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393, Montreal, Canada.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 Shared Task: Semantic Textual Similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 32–43, Atlanta, Georgia, USA.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 81–91, Dublin, Ireland.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 252–263, Denver, Colorado, USA.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, Baltimore, Maryland, USA.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An Algorithm that Learns what's in a Name. *Machine Learning*, 34 (1-3): 211–231.

Chris Brockett. 2007. Aligning the RTE 2006 Corpus. Technical Report MSR-TR-2007-77, Microsoft Research.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31 (3), pages 59–79.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 758–764, Atlanta, Georgia, USA.

Michael Heilman and Noah A. Smith 2010. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1011–1019, Los Angeles, California, USA.

Adam Lally, John M. Prager, Michael C. McCord, Branimir K. Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question Analysis: How Watson Reads a Clue. *IBM Journal of Research and Development*, 56 (3.4): 2:1–2:14.

Jimmy Lin and Boris Katz. 2006. Building a Reusable Test Collection for Question Answering. *Journal of the American Society for Information Science and Technology*, 57 (7): 851–861.

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*, Ann Arbor, Michigan, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations Workshop*, Scottsdale, Arizona, USA.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, pages 2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, Pennsylvania, USA.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 458–467, Seattle, Washington, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, Santiago, Chile.

Eyal Shnarch. 2013. Probabilistic Models for Lexical Inference. *PhD Thesis*, Bar Ilan University.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to Basics for Monolingual Alignment: Exploiting Word Similarity and Contextual Evidence. *Transactions of the Association for Computational Linguistics*, 2, pages 219–230.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 148–153, Denver, Colorado, 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*, pages 22–32, Prague, Czech Republic.

Mengqiu Wang, and Christopher D. Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1164–1172, Beijing, China.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013a. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867, Atlanta, Georgia, USA.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013b. Semi-Markov Phrase-Based Monolingual Alignment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 590–600, Seattle, Washington, USA.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013c. Automatic Coupling of Answer Extraction and Information Retrieval. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 159–165, Sofia, Bulgaria.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1744–1753, Sofia, Bulgaria.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *Proceedings of the Deep Learning and Representation Learning Workshop, NIPS 2014*, Montréal, Canada.