Higher-order Lexical Semantic Models for Non-factoid Answer Reranking

Daniel Fried[1], Peter Jansen[1], Gustave Hahn-Powell[1], Mihai Surdeanu[1], and **Peter Clark[2]**
[1] University of Arizona, Tucson, AZ, USA
[2] Allen Institute for Artificial Intelligence, Seattle, WA, USA
{dfried,pajansen,hahnpowell,msurdeanu}@email.arizona.edu
peterc@allenai.org

Abstract

Lexical semantic models provide robust performance for question answering, but, in general, can only capitalize on direct evidence seen during training. For example, monolingual alignment models acquire term alignment probabilities from semi-structured data such as question-answer pairs; neural network language models learn term embeddings from unstructured text. All this knowledge is then used to estimate the semantic similarity between question and answer candidates. We introduce a higher-order formalism that allows all these lexical semantic models to chain direct evidence to construct indirect associations between question and answer texts, by casting the task as the traversal of graphs that encode direct term associations. Using a corpus of 10,000 questions from Yahoo! Answers, we experimentally demonstrate that higher-order methods are broadly applicable to alignment and language models, across both word and syntactic representations. We show that an important criterion for success is controlling for the semantic drift that accumulates during graph traversal. All in all, the proposed higher-order approach improves five out of the six lexical semantic models investigated, with relative gains of up to +13% over their first-order variants.

1 Introduction

Open-domain question answering (QA), which finds short textual answers to natural language questions, is often viewed as the successor to keyword search (Etzioni, 2011) and one of the most difficult and widely applicable end-user applications of natural language processing (NLP). From syntactic parsing, discourse processing, and lexical semantics, QA necessitates a level of functionality across a variety of topics that make it a natural, yet challenging, proving ground for many aspects of NLP. Here, we address a particularly challenging QA subtask: open-domain non-factoid QA, where queries take the form of complex questions (e.g., manner or *How* questions), and answers range from single sentences to entire paragraphs. Because this task is so complex and large in scope, current state-of-the-art open-domain systems perform at only about 30% P@1, or answering roughly one out of three questions correctly (Jansen et al., 2014).

In this paper we focus on answer ranking (AR), a key component of non-factoid QA that focuses on ordering candidate answers based on the likelihood that they capture the information needed to answer a question. Unlike keyword search, Berger (2000) observed that lexical matching methods are generally insufficient for QA, where questions and answers often have little to no lexical overlap (as in the case of *Where should we go for breakfast?* and *Zoe's Diner has great pancakes*). Previous work has shown that lexical semantics (LS) models are well suited to bridging this "lexical chasm", and at least two flavors of lexical semantics have been successfully applied to QA. The first treats QA as a monolingual alignment problem, learning associations between words (or other structures) that appear in question-answer pairs (Surdeanu et al., 2011; Yao et al., 2013). The second computes the semantic similarity between question and answer using language models acquired from relevant texts (Yih et al., 2013; Jansen et al., 2014).

Here we argue that while these models begin to bridge the "lexical chasm", many still suffer from sparsity and only capitalize on direct evidence. Returning to our example question, if we also train on the QA pair *What goes well with pancakes?* and *hashbrowns and toast*, we can use the

Transactions of the Association for Computational Linguistics, vol. 3, pp. 197–210, 2015. Action Editor: Sharon Goldwater.
Submission batch: 12/2014; Revision batch 3/2015; Published 4/2015.

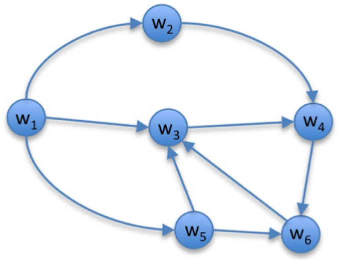

Figure 1: A hypothetical graph derived from a lexical semantic alignment model.

indirect association between *breakfast – pancakes* and *pancakes – hashbrowns* to locate additional answers for where to visit for breakfast, including *Regee's has the best hashbrowns in town*.

We can represent LS models as a graph, as in Figure 1. For a word-based alignment model, the graph nodes represent individual words, and the (directed) edges capture the likelihood of a word $\mathbf{w}_a$ appearing in a gold answer given word $\mathbf{w}_q$ in the question. Grounding this in our example, $\mathbf{w}_1$ may represent *breakfast*, $\mathbf{w}_2$ *pancakes*, and $\mathbf{w}_4$ *hashbrowns*. For a language model, the edges capture the similarity of contexts between the words in the nodes. For both alignment and language model graphs, we observe that semantic association between two words (or structures) stored in the nodes can be determined by investigating the different paths that connect these two nodes, even when they are not directly connected.

We call this class of models *higher-order lexical models*, in contrast to the *first-order lexical models* introduced in previous work, which rely only on direct evidence to estimate association strength. For example, all alignment models in previous work can estimate $P(\mathbf{w}_q|\mathbf{w}_a)$, i.e., the probability of generating the question word $\mathbf{w}_q$ if the answer contains the word $\mathbf{w}_a$, only if this pair has been seen at least once in training. On the other hand, our approach can estimate $P(\mathbf{w}_q|\mathbf{w}_a)$ from indirect evidence, e.g., from chaining two distinct question/answer pairs that contain the words ($\mathbf{w}_q$, $\mathbf{w}_i$) and ($\mathbf{w}_i$, $\mathbf{w}_a$), respectively.

The contributions of this work are:

1. This is the first work to our knowledge that proposes higher-order LS models for QA. We show that an important criterion for success is controlling for the semantic drift that accumulates in the graph traversal paths, which plagues traditional random walk algorithms. For example, we empirically demonstrate that paths up to a length of three are generally beneficial, but longer paths hurt or do not improve performance.
2. We show that a variety of LS models and representations, including alignment and language models, over both words and syntactic structures, can be adapted to the proposed higher-order formalism. In this latter respect we introduce a novel syntax-based variant of the neural network language model (NNLM) of Mikolov et al. (2013) that models syntactic dependencies rather than words, which allows it to capture knowledge that is complementary to that of word-based NNLMs.
3. The training process for alignment models requires a large corpus of QA pairs. Due to these resource requirements, we evaluate our higher-order LS models on a community question answering (CQA) task (Wang et al., 2009; Jansen et al., 2014) across thousands of *how* questions, and show that most higher-order models perform significantly better than their first-order variants.
4. We demonstrate that language models and alignment models capture complementary information, and can be combined to improve the performance of the CQA system for manner questions.

2 Related Work

We focus on statistical LS methods for open-domain QA, in particular CQA tasks, such as the ones driven by Yahoo! Answers datasets (Wang et al., 2009; Jansen et al., 2014).

Berger et al. (2000) were the first to propose a statistical LS model to "bridge the lexical chasm" between questions and answers for a QA task. Building off this work, a number of LS models using either words or other syntactic and semantic structures have been proposed for QA (Echihabi and Marcu, 2003; Soricut and Brill, 2006; Riezler et al., 2007; Surdeanu et al., 2011; Yao et al., 2013; Jansen et al., 2014), or related tasks, such as semantic textual similarity (Sultan et al., 2014). However, all of these models fall into the class of *first-order* models. To our knowledge, this work is the first to investigate higher-order LS models for QA, which can take advantage of indirect evidence, i.e., the "neighbors of neighbors" in the association graph.

Second-order lexical models have recently been brought to bear on a variety of other NLP tasks. Zapirain et al. (2013) use a second-order model based on distributional similarity (Lin, 1998) to improve a selectional preference model, and Lee et al. (2012) use a similar approach for coreference resolution. Our work expands on these ideas with models of arbitrary order, an approach to control semantic drift, and an application to QA.

This work falls under the larger umbrella of algorithms for graph-based inference, which have been successfully applied to other NLP and information retrieval problems, such as relation extraction (Chakrabarti and Agarwal, 2006; Chakrabarti, 2007; Lao and Cohen, 2010), inference over knowledge bases (Lao et al., 2011), name disambiguation (Minkov et al., 2006), and search (Bhalotia et al., 2002; Tong et al., 2006). While many of these approaches use random walk algorithms, in pilot experiments we observed that random walks, such as PageRank (PR) (Page et al., 1999), tend to accumulate semantic drift from the originating node because they consider all possible paths in the graph. This semantic drift reduces the quality of the higher-order associations, which impacts QA performance. Here we implement a *conservative* graph traversal algorithm, similar in spirit to the "cautious bootstrapping" algorithm of Yarowsky (Whitney and Sarkar, 2012; Yarowsky, 1995). By constraining the traversal paths, our algorithm runs two orders of magnitude faster than PR, while controlling for semantic drift.

3 Approach

The architecture of our proposed QA framework is illustrated in Figure 2. Here we evaluate both first-order and higher-order LS models in the context of community question answering (CQA), using a large dataset of QA pairs from Yahoo! Answers[1]. We use a standard CQA evaluation task (Jansen et al., 2014), where one must rank a set of user-generated answers to a given question, such that the community-selected best answer appears in the top position. More specifically, for a given question, the candidate retrieval (CR) component fetches all its community-generated answers from the answer collection, and gives each answer candidate an initial ranking using shallow information retrieval scoring.[2] These answer candidates are then passed on to the answer reranking component, the focus of this work.

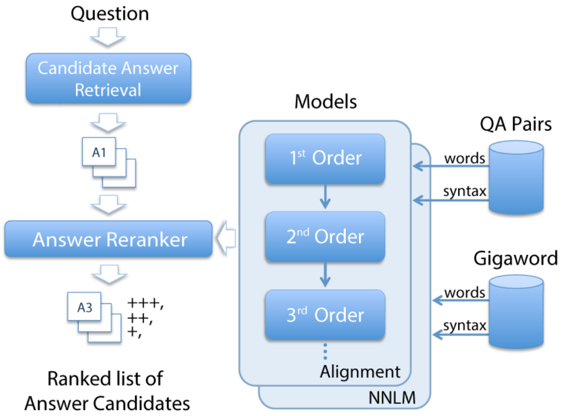

Figure 2: Architecture of the reranking framework for QA, showing a reranking component that incorporates two classes of lexical semantic models (alignment and neural network language models), implemented over two representations of content (words and syntax).

The answer reranking (AR) component analyzes the answer candidates using more expensive techniques to extract first-order and higher-order LS features, and these features are then used in concert with a learning framework to rerank the candidates and elevate correct answers to higher positions. For the learning framework, we use SVM^{rank}, a variant of Support Vector Machines for structured output adapted to ranking problems.[3] In addition to these features, each reranker also includes a single feature containing the score of each candidate, as computed by the above candidate retrieval component.[4]

4 First-Order Lexical Models

We next introduce a series of first-order LS models, which serve as the foundation for the higher-order models discussed in the next section.

4.1 Neural Network Language Models

Inspired by previous work (Yih et al., 2013; Jansen et al., 2014), we adapt the `word2vec` NNLM of

[1] `http://answers.yahoo.com`

[2] We used the same scoring as Jansen et al. (2014): cosine similarity between the question and candidate answer's lemma vector representations, with lemmas weighted using *tf.idf* (Ch. 6, (Manning et al., 2008)).

[3] `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

[4] Including these scores as features in the reranker model is a common strategy that ensures that the reranker takes advantage of the analysis already performed by the CR model.

Mikolov et al. (2013) to this QA task. In particular, we use their skip-gram model with hierarchical sampling. This model predicts the context of a word given the word itself, and through this process embeds words into a latent conceptual space with a fixed number of dimensions. Consequently, related words tend to have vector representations that are close to each other in this space. This type of predictive algorithm has been found to perform considerably better than count-based approaches to distributional similarity on a variety of semantic tasks (Baroni et al., 2014).

We derive four LS measures from these vectors, which are then are included as features in the reranker. The first is a measure of the **overall similarity** of the question and answer candidate, which is computed as the cosine similarity between two composite vectors. These composite vectors are assembled by summing the vectors for individual question (or answer candidate) words, and re-normalizing this composite vector to unit length.[5] In addition to this overall similarity score, we compute the pairwise similarities between each word in the question and answer candidates, and include as features the **average, minimum, and maximum pairwise similarities**.

4.2 Alignment Models

Berger et al. (2000) showed that learning question-to-answer transformations using a statistical machine translation (SMT) model begins to "bridge the lexical chasm" between questions and answers. We build upon this observation, using the IBM Model 1 (Brown et al., 1993) variant of Surdeanu et al. (2011) to determine the probability that a question Q is a translation of an answer A, $P(Q|A)$:

$$P(Q|A) = \prod_{q \in Q} P(q|A) \quad (1)$$

$$P(q|A) = (1-\lambda)P_{ml}(q|A) + \lambda P_{ml}(q|\mathbf{C}) \quad (2)$$

$$P_{ml}(q|A) = \sum_{a \in A} (T(q|a)P_{ml}(a|A)) \quad (3)$$

where the probability that the question term q is generated from answer A, $P(q|A)$, is smoothed using the prior probability that the term q is generated from the entire collection of answers $\mathbf{C}$, $P_{ml}(q|\mathbf{C})$. $P_{ml}(q|A)$ is computed as the sum of the probabilities that the question term q is a translation of any answer term a, $T(q|a)$, weighted by the probability that a is generated from A. The translation table for $T(q|a)$ is computed using GIZA++ (Och and Ney, 2003).

Similar to Surdeanu et al. (2011) we: (a) set $P_{ml}(q|\mathbf{C})$ to a small value for out-of-vocabulary words; (b) modify the $T(.|.)$ distributions to guarantee that the probability of translating a word to itself, i.e., $T(w|w)$, is highest (Murdock and Croft, 2005); and (c) tune the smoothing parameter λ on a development corpus.

QA systems generally use the above model to determine the **global alignment probability** between a given question and answer candidate, $P(Q|A)$. A novel contribution of our work is that we also use the alignment model's probability distributions (from a source word to destination words) as distributed representations for source words, based on the observation that words with similar alignment distributions are likely to have a similar meaning. Formally, we denote the alignment vector representation for the ith word in the vocabulary as $\mathbf{w}(i)$, and define the jth entry in the vector as $\mathbf{w}(i)_j = T(word_j|word_i)$, where $T(.|.)$ is the translation probability distribution from Eq. 3. That is, the jth entry of the vector for word i is the conditional probability of seeing $word_j$ in a question, given that $word_i$ was seen in a corresponding answer. The vector for each word then represents a discrete (conditional) probability distribution. To compare two words, we use the square root of the Jensen-Shannon divergence (JSD) between their conditional distributions.[6] Let $\mathbf{m}(i,j)$ be the element-wise average of the vectors $\mathbf{w}(i)$ and $\mathbf{w}(j)$, that is $(\mathbf{w}(i) + \mathbf{w}(j))/2$, and let $K(\mathbf{w}, \mathbf{v})$ be the Kullback-Leibler divergence between distributions (represented as vectors) $\mathbf{w}$ and $\mathbf{v}$:

$$K(\mathbf{w}, \mathbf{v}) = \sum_{i=0}^{|V|} \mathbf{w}_i \ln \frac{\mathbf{w}_i}{\mathbf{v}_i} \quad (4)$$

where $|V|$ is the vocabulary size (and the dimensionality of $\mathbf{w}$). Then the distance between words i and j is

$$J(\mathbf{w}(i), \mathbf{w}(j)) = \sqrt{\frac{K(\mathbf{w}(i), \mathbf{m}(i,j)) + K(\mathbf{w}(j), \mathbf{m}(i,j))}{2}} \quad (5)$$

[5] We also tried the multiplicative strategy for word-vector combination of Levy and Goldberg (2014b), but it did not improve our results.

[6] We use the square root of the Jensen-Shannon divergence, derived from Kullback-Leibler divergence, since it is a distance metric (in particular it is finite and symmetric).

We derive four additional LS features from these alignment vector representations, which parallel the features derived from the NNLM vector representations (§4.1). The first is the **JSD between the composite alignment vectors** for the question and the answer candidate, where the composite is constructed by summing the vectors for individual question (or answer candidate) words, and dividing by the number of vectors summed. We also generate features from the **average, minimum, and maximum pairwise JSD between words** in the question and words in the answer candidate. In practice, we have found that the addition of these four features considerably improves the performance of all alignment models investigated for the QA task.

4.3 Modeling Syntactic Structures

Surdeanu et al. (2011) demonstrated that alignment models can readily capitalize on representations other than words, including representations of syntax. Here we expand on their work and explore three syntactic variations of the alignment and NNLM models. For these models we make use of collapsed syntactic dependencies with processed CC complements (de Marneffe et al., 2006), extracted from the Annotated English Gigaword (Napoles et al., 2012).

The first model is a straightforward adaptation of the alignment model in §4.2, where the representations of questions and answers are changed from bags of words to "bags of syntactic dependencies" (Surdeanu et al., 2011). That is, the terms to be aligned (q and a in Eq. 1 to 3) become $\langle head, label, dependent\rangle$ dependencies, such as $\langle eat, dobj, pancakes\rangle$, rather than words. The motivation behind this change is that it allows the model to align simple propositions rather than words, which are a more accurate representation of the information encoded in the question. While tuning on the development set of questions we found that unlabeled dependencies, which ignore the middle term in the above tuples, performed better than labeled dependencies, possibly due to the sparsity of the labeled dependencies. As such, all our dependency alignment models use unlabeled dependencies.

The second model adapts the idea of working with bags of syntactic dependencies, instead of bags of words, to NNLMs, and builds an LM over dependencies rather than words. However, because the NNLM embeddings are generated from context, we must find a consistent ordering of dependencies that maintains contextual information. Here we order dependencies using a depth-first, left-to-right traversal of the dependency graph originating at the predicate or root node.[7] Figure 3 shows an example of such a traversal, which results in the following ordering (using unlabeled dependencies):

(6) be_time time_the time_same be_country country_no
be_should be_business business_the
business_concealing concealing_history history_its

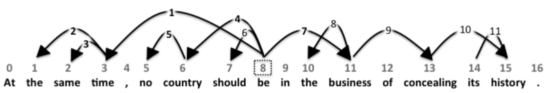

Figure 3: DFS L→R ordering of word pairs starting from the sentential root. The sentence is taken from `LTW_ENG_20081201.0002` of the English Gigaword.

The model uses the default `word2vec` implementation to compute the term embeddings. To our knowledge, this syntax-driven extension of NNLMs is novel.

During tuning we compared the labeled and unlabeled dependency representations against an NNLM model using traditional bigrams (where words are paired based on their linear order, i.e., without any syntactic motivation). Again, we found that unlabeled dependencies outperformed surface bigrams and labeled dependencies.

Lastly, we also evaluate the recent NNLM model proposed by Levy and Goldberg (2014a), who modified the `word2vec` embedding procedure to include context derived from dependencies, arguing that this embedding model encodes functional similarity better than Mikolov et al.'s unigram surface structure model. It is important to note that while in our syntactic NNLM model the LM terms are (lexicalized unlabeled) syntactic dependency tuples, in Levy and Goldberg's model the LM terms are still words, but the *context* is driven by syntax instead of surface structure.

5 Higher-order lexical models

The first-order models we have just described make use of the direct evidence seen in training data to create associations between words (or dependencies). Unfortunately this training data is often quite sparse, and while it may be possible to exhaustively create training data with broad

[7] Other ordering schemes were tested during development, but DFS proved to be the most stable across sentences.

coverage for artificially narrow domains such as *things to eat for breakfast*, this quickly becomes intractable for larger domains. To mitigate this sparsity, here we propose a method that identifies associations that were not explicitly seen during training. Returning to our example in §1, had an alignment model learned the associations *breakfast – pancakes* and *pancakes – hashbrowns*, the model should be able to make use of the indirect association between *breakfast – hashbrowns* to further "bridge the lexical chasm", and identify answer candidates containing the word *hashbrowns* to the question *Where is a good place to eat breakfast?*

At first glance, algorithms such as PageRank (Page et al., 1999) appear to offer an attractive and intuitive method to determine the association between arbitrary nodes in a weighted graph by modeling a random walk to convergence (from any given starting node). As we show in Section 6, we have found that these methods quickly accumulate semantic drift (McIntosh, 2010), because: (a) they run to convergence, i.e., they traverse paths of arbitrary length, and (b) they explore the entire association space encoded in the neighborhood graph, both of which introduce considerable noise. For example, by chaining even small numbers of direct associations, such as *breakfast – pancakes*, *pancakes – hashbrowns*, *hashbrowns – potato*, and *potato – field*, a model that does not control for semantic drift may be tempted to answer questions about breakfast venues with answers that discuss *wheat fields* or *soccer fields*. Thus, keeping these indirect associations short and on-context is critical for QA.

Based on the above observations, here we outline a general method for creating higher-order lexical semantic models that incorporate simple measures to control for semantic drift, and demonstrate how to apply it to both alignment graph representations and the distributed vector space representations of a NNLM. Our method relies on two general properties of the first-order LS models: (a) all models include a **distributed representation** for each term[8] that is encoded as a vector (e.g., for NNLMs this is a term's embedding in vector space, and for alignment models this is a vector of alignment probabilities from a given answer term to all possible question terms), and (b) models define a pairwise function that assigns a **lexical similarity** score between pairs of directly connected terms (e.g., for NNLMs this is the cosine similarity of the corresponding term embeddings, and for alignment models this is given directly by $T(q|a)$, from Equation 3).

Intuitively, we incorporate indirect evidence in our models by averaging the distributed representation of a term t with the distributed representations of its neighbors, i.e., those terms with highest lexical similarity to t. To control for semantic drift we consider: (a) only the k most similar neighbors to the current term, and (b) short paths in the association graph, e.g., neighbors of neighbors but no further. For example, for a second-order model, we construct this new vector representation for each node in the graph by averaging the vector of the node itself and the vectors of its k closest neighbors (weighted by each pair's similarity), so that the vector representation of *breakfast* becomes a weighted sum of *breakfast, morning, food, cereal,* and so forth. By using only the closest neighbors, we exclude low associates of *breakfast*, such as *spoon* or *vacation*, which are only marginally related and therefore likely to generate semantic drift.

Formally, let $\mathbf{w}$ be the vector representation (either from an NNLM or alignment model) of a given term. We construct a *higher-order* representation $\hat{\mathbf{w}}$ by linearly interpolating $\mathbf{w}$ with its *top associates*, those vectors with highest value of a pairwise scoring function s:

$$\hat{\mathbf{w}}(i) = \sum_{j \in N_k(\mathbf{w}(i))} s(\mathbf{w}(i), \mathbf{w}(j))\, \mathbf{w}(j) \qquad (7)$$

where s is the function that measures the lexical similarity between terms, and depends on the representations used (s_{align} or s_{cos}, see below). $N_k(\mathbf{w}(i))$, the *neighborhood* of the vector $\mathbf{w}(i)$, denotes the indices of the k term vectors with highest values of $s(\mathbf{w}(i), \cdot)$, and k is a hyperparameter that controls how many vectors are interpolated. The resulting vectors are renormalized, as discussed later.[9]

It is important to note that this process of creating higher-order vectors can be trivially

[8] Henceforth we use "term" to indicate either a word or a dependency in a lexical semantics model.

[9] $N_k(\mathbf{w})$ includes $\mathbf{w}$, because in all LS models proposed here, each term has maximal similarity with itself. The normalization applied after combining all vectors has the effect of modulating the contribution of the vector $\mathbf{w}$ in the higher order representation $\hat{\mathbf{w}}$, based on how similar the other vectors in $N_k(\mathbf{w})$ are to $\mathbf{w}$.

iterated arbitrarily, constructing a second-order model from a first-order model, then a third-order model from a second, etc. Intuitively, this models paths of increasing length in the association graph over terms. We now describe the concrete implementations of the procedure in the alignment and the NNLM settings.

Higher-order Alignment: In the alignment setting, where the vector $\mathbf{w}(i)$ for term i contains the probabilities that the source (answer) term i is aligned with the destination (question) term j, we use

$$s_{align}(\mathbf{w}(i), \mathbf{w}(j)) = T(term_j|term_i) \tag{8}$$

where T is the translation probability from Equation (3). We normalize the $\hat{\mathbf{w}}$ vectors of Equation (7) using the L_1 norm, so that each continues to represent a probability distribution. Intuitively, the second-order alignment distribution $\hat{\mathbf{w}}(i)$ for source term i is a linear combination of term i's most probable destination terms' distributions, weighted by the destination probabilities from $\mathbf{w}(i)$.

The higher-order models greatly reduce the sparsity of the first-order alignment graph. In a first-order word alignment model (i.e., using only direct evidence) trained on approximately 70,000 question/answer pairs (see §6.1), source words align with an average of 31 destination words with non-zero probability, out of a 103,797 word vocabulary. The second-order model aligns 1,535 destination words to each source word on average; the third-order model, 8,287; and the fourth-order model, 15,736. In the results section, we observe that this sparsity reduction is only helpful up to a certain point; having too many associates for each word reduces performance on the QA task.

Higher-order NNLM: In the NNLM setting, we use the cosine similarity of vectors as interpolation weights and to choose the nearest neighbors:

$$s_{cos}(\mathbf{w}(i), \mathbf{w}(j)) = \frac{\mathbf{w}(i) \cdot \mathbf{w}(j)}{||\mathbf{w}(i)||\,||\mathbf{w}(j)||} \tag{9}$$

We found that applying the softmax function to each term's vector of k-highest s_{cos} similarities, to ensure all interpolation weights are positive and have a consistent range across terms, improved performance. As such, all higher-order NNLM models use this softmax normalization.

The resulting interpolation can be conceptualized in several ways. Viewing cosine similarity as a representation of entailment (Beltagy et al., 2013), the higher-order NNLM model reflects multiple-hop inference on top of the corresponding association graph, similar to the higher-order alignment model. The interpolation could also be viewed as smoothing term representations in vector space, averaging each term's vector with its nearest neighbors according to their cosine similarity.

Higher-order Hybrid Model: We also implement a *hybrid model*, which interpolates the alignment distribution vectors, but using the pairwise cosine similarities from the NNLM setting:

$$\hat{\mathbf{w}}_{\mathbf{A}}(i) = \sum_{j \in N_k(\mathbf{w}_{\mathbf{N}}(i))} s_{cos}(\mathbf{w}_{\mathbf{N}}(i), \mathbf{w}_{\mathbf{N}}(j))\, \mathbf{w}_{\mathbf{A}}(j)$$

where $\mathbf{w}_{\mathbf{N}}(i)$ and $\mathbf{w}_{\mathbf{A}}(i)$ are respectively the NNLM vector and alignment vector representations for term i, and s_{cos} is cosine similarity. In addition, we experimented with the opposite hybrid model: interpolating the NNLM vectors using alignment associate probabilities as weights, but found that it did not perform as well as using NNLM similarities to interpolate alignment vectors. Our conjecture is that the higher order technique can be viewed as a method of reducing sparsity (either in the vectors or in the context used to create them), and since the NNLM vectors trained on words (as opposed to dependencies) are less sparse, they benefit less from this technique.

6 Experiments

6.1 Data

To test the utility of our approach, we experimented with the QA scenario introduced in §3 using the subset of Yahoo! Answers Corpus introduced by Jansen et al. (2014)[10]. Yahoo! Answers[11] is an open domain community-generated QA site, with questions and answers that span formal and precise to informal and ambiguous language. This corpus contains 10,000 QA pairs from a corpus of *how* questions. Each question contains at least four community-generated answers, one of which was voted as the top answer. The number of answers to each question ranged from 4 to over 50,

[10] http://nlp.sista.arizona.edu/releases/acl2014

[11] http://answers.yahoo.com

with the average 9. 50% of QA pairs were used for training, 25% for development, and 25% for test.

The following additional resources were used:

NNLM Corpus: We generated vector representations for words using the `word2vec` model of Mikolov et al. (2013), using the skip-gram architecture with hierarchical sampling. Vectors are trained using the entire Gigaword corpus of approximately 4G words.[12] We used 200-dimensional vectors (Jansen et al., 2014).

Alignment Corpus: We use a separate partition of QA pairs to train an alignment model between questions and top answers. This corpus of 67,972 QA pairs does not overlap with the collection of 10,000 pairs used for the training, development, and testing partitions, but was chosen according to the same criteria. The alignment model was trained using GIZA++ over five iterations of IBM Model 1.

6.2 Tuning

The following hyperparameters were tuned independently to maximize P@1 on the development partition of the Yahoo! Answers dataset:

Number of Vectors Interpolated: We investigated the effects of varying k in Eq. 7, i.e., the number of most-similar neighbor vectors interpolated when constructing a higher-order model. We experimented with values of k ranging from 1 to 100 and found that the value does not substantially affect results across the second-order word alignment models. Since vectors used in the higher-order interpolation are weighted by their similarity to a vector $\mathbf{w}$, this is likely due to a swift decrease in values of the pairwise similarity function, $s(\mathbf{w}, \cdot)$, beyond the vectors closest to $\mathbf{w}$. We chose to use $k = 20$ because it comes from a stabler maximum, and it is sufficiently small to make construction of higher-order models feasible in terms of time and memory usage.

Base Representation Type: We compared the effects of using either words or lemmas as the base lexical unit for the LS models, and found that words achieved higher P@1 scores in both the alignment and NNLM models on the development dataset. As such, all results reported here use words for the syntax-independent models, and tuples of words for the syntax-driven models.

Content Filtering: We investigated using part-of-speech (POS) tags to filter the content considered by the lexical similarity models, by excluding certain non-informative classes of words such as determiners. Using POS tags generated by Stanford's CoreNLP (Manning et al., 2014), we filtered content to only include nouns, adjectives, verbs, and adverbs for the word-based models, and tuples where *both* words have one of these four POS tags for the syntax-based models. We found that this increased P@1 scores for all word-based alignment and NNLM models (including the Levy-Goldberg (L-G) model[13]), but did not improve performance for models that used dependency representations.[14] Results reported in the remainder of this paper use this POS filtering for all word-based alignment and NNLM models (including L-G's) as well as the dependency alignment model, but not for our dependency NNLM model.[15]

6.3 Baselines

We compare against five baseline models:

Random: selects an answer randomly;

CR: uses the ranking provided by our shallow answer Candidate Retrieval (CR) component;

Jansen et al. (2014): the best-performing lexical semantic model reported in Jansen et al. (2014) (row 6 in their Table 1).[16]

PR$_\mathrm{U}$: PageRank (Page et al., 1999) with uniform teleportation probabilities, constructed over the word alignment graph. Let A be the row-stochastic transition matrix of this graph, where the ith row of A is the vector $\mathbf{w}(i)$ (see §4.2). Following the PR algorithm, we add small "teleportation" probabilities from a teleportation matrix T[17] to the alignment matrix, producing a PageRank

[12] LDC catalog number LDC2012T21

[13] We modified the L-G algorithm to ignore any dependencies not matching this filtering criterion when building the context for a word.

[14] We hypothesize that our filtering criterion for dependencies, which requires both head and modifiers to have one of the four POS tags, was too aggressive. We will explore other filtering criteria in future work.

[15] Although filtered dependencies yielded slightly lower results in our tuning experiments, we use them in the dependency alignment experiments because the unfiltered third-order dependency alignment model was too large to fit into memory.

[16] We do not report the performance of their discourse-based models, because these models use additional resources making them incompatible with the work shown here.

[17] We construct the teleportation matrix T by setting each row of T to be a uniform probability distribution, i.e., each entry in a row of T is $1/|V|$, where $|V|$ is the vocabulary size.

matrix $P = \alpha * A + (1 - \alpha) * T$. Following previous work (Page et al., 1999), we used $\alpha = 0.15$. Then, we compute the matrix products $P^2, P^3, ...$, using rows in these products as vector representations for terms. Thus, the ith row of P^k gives the conditional probabilities of arriving at each term in the graph after k transitions beginning at term i.

PR$_\mathbf{P}$: Personalized PageRank (Agirre and Soroa, 2009). This method follows the same algorithm as PR$_\mathrm{U}$, but uses NNLM similarities of terms to define non-uniform teleportation probabilities (following the intuition that teleportation likelihood should be correlated with semantic similarity of terms). In particular, we obtain the ith row of T by computing the pairwise cosine similarities of term i's NNLM vector with every other term's vector, and taking the softmax of these similarities to produce a probability distribution, $\mathbf{v}(i)$. To account for missing terms (due to the difference in the alignment and NNLM corpora) we smooth all teleportation probabilities by adding a small value, ϵ, to each entry of $\mathbf{v}(i)$. The renormalized $\mathbf{v}(i)$ then becomes the ith row of T.

6.4 Results

Analysis of higher-order models

Table 1 shows the performance of *first-order* and *higher-order* alignment and NNLM models across representation type. Rows in the table are labeled with the orders of the representations used. W indicates a model where the base units are words, while D indicates a model where the base units are dependencies; A denotes an alignment model, while N denotes an NNLM model. Distinct features are generated for each order representation (§§4.1, 4.2, 5) and used in the SVM model. For example, WN(1-3) uses features constructed from the 1st, 2nd, and 3rd order representations of the word-based NNLM. Combining multiple orders into a single feature space is important, as each order captures different information. For example, the top five closest associates for the term *dog* under WN(1) are: *puppy*, *cat*, *dogs*, *dachshund*, and *pooch*, whereas the top associates under WN(2) are: *mixedbreed*, *hound*, *puppy*, *rottweiler*, and *groomer*. Aggregating features from all orders into a single ranking model guarantees that all this information is jointly modeled.

We use the standard implementations for precision at 1 (P@1) and mean reciprocal rank (MRR) (Manning et al., 2008).

Several trends are apparent in the table:

(i) All first-order lexical models outperform the random and CR baselines, as well as the system of Jansen et al. The latter result is explained by the novel features proposed in §4.1 and §4.2. We detail the contribution of these features in the ablation experiment summarized later in Table 4.

(ii) More importantly, both P@1 and MRR generally increase as we incorporate higher-order versions of each lexical model. Out of the six LS models investigated, five improve under their corresponding higher-order configurations. This gain is most pronounced for the dependency alignment (DA) model, whose P@1 increases from 25.9% for the first-order model to 29.4% for a model that incorporates first, second, and third-order features. In general, the performance increase in higher-order models is larger for sparser first-order models, where the higher-order models fill in the knowledge gaps of their corresponding first-order variants. This sparsity is generated by multiple causes: (a) alignment models are sparser than NNLMs because they were trained on less data (approximately 67K question-answer pairs vs. the entire Gigaword); (b) models that use syntactic dependencies as the base lexical units are sparser than models that use words. We conjecture that this is why we see the biggest improvement from higher order for DA (which combines both sources of sparsity), and we do not see an improvement for the word-based NNLM (WN). This hypothesis is supported by the analysis of the corresponding association graphs: in the WA model, counting the number of words comprising the top 99% of the probability mass distribution for each word shows sparse representations with 10.1 most-similar words on average; in the WN model, the same statistic shows dense distributed representations with the top 99% of mass distributed across 1.2 million most-similar words per word (or, about 99% of the vocabulary).

(iii) The higher-order models perform well up to an order of 3 or 4, but not further. For example, the performance of the word alignment model (WA) decreases at order 4 (row 7); similarly, the performance of the word NNLM (WN) decreases at order 5 (row 12). This supports our initial observation that long paths in the association graph accumulate noise, which leads to semantic drift.

(iv) The Levy-Goldberg NNLM (DN$_\text{L-G}$) is the best first-order model, whereas our dependency-

#	Model	P@1	P@1 Impr.	MRR	MRR Impr.
	Baselines				
1	Random	14.29		26.12	
2	CR	19.57	–	43.15	–
3	Jansen et al. (2014)	26.57	–	49.31	–
	Word Alignment (WA)				
4	CR + WA(1)	27.33	+40%	49.20	+14%
5	CR + WA(1-2)	29.01*	+48%	50.87*	+18%
6	CR + WA(1-3)	**30.49***	**+56%**	**51.92***	**+20%**
7	CR + WA(1-4)	29.65*	+52%	51.38*	+19%
	Word NNLM (WN)				
8	CR + WN(1)	**30.69**	**+57%**	**52.02**	**+21%**
9	CR + WN(1-2)	29.57	+51%	51.29	+19%
10	CR + WN(1-3)	30.21	+54%	51.75	+20%
11	CR + WN(1-4)	30.41	+55%	51.91	+20%
12	CR + WN(1-5)	30.05	+54%	51.87	+20%
	Dependency Alignment (DA)				
13	CR + DA(1)	25.89	+32%	48.24	+12%
14	CR + DA(1-2)	28.81*	+47%	50.50*	+17%
15	CR + DA(1-3)	**29.41***	**+50%**	**51.14***	**+19%**
	Our Dependency NNLM (DN$_{\text{O}}$)				
16	CR + DN$_{\text{O}}$(1)	30.85	+58%	51.93	+20%
17	CR + DN$_{\text{O}}$(1-2)	31.69*	+62%	52.35*	+21%
18	CR + DN$_{\text{O}}$(1-3)	**31.89***	**+63%**	**52.43***	**+22%**
	Levy-Goldberg Dependency NNLM (DN$_{\text{L-G}}$)				
19	CR + DN$_{\text{L-G}}$(1)	31.25	+60%	52.60	+22%
20	CR + DN$_{\text{L-G}}$(1-2)	31.57	+61%	**52.76**	**+22%**
21	CR + DN$_{\text{L-G}}$(1-3)	**31.69**	**+62%**	52.73	+22%
	Hybrid Model: Word Alignment with NNLM Similarity (WA$_{\text{N}}$)				
22	CR + WA$_{\text{N}}$(1-2)	29.37	+50%	51.05	+18%
23	CR + WA$_{\text{N}}$(1-3)	**30.45***	**+56%**	**51.84***	**+20%**
	PageRank Model: Uniform Teleportation (PR$_{\text{U}}$)				
24	CR + PR$_{\text{U}}$(1)	27.29	+39%	49.21	+14%
25	CR + PR$_{\text{U}}$(1-2)	**30.33***	**+55%**	51.74*	+20%
26	CR + PR$_{\text{U}}$(1-3)	30.17*	+54%	**51.87***	**+20%**
	PageRank Model: Personalized Teleportation (PR$_{\text{P}}$)				
27	CR + PR$_{\text{P}}$(1)	27.09	+38%	49.10	+14%
28	CR + PR$_{\text{P}}$(1-2)	**31.01***	**+58%**	**52.25***	**+21%**
29	CR + PR$_{\text{P}}$(1-3)	29.89*	+53%	51.70*	+20%

Table 1: Overall results on the Yahoo! Answers dataset using first-order representations (1) and first-order in combination with higher-order representations (1-*n*). Bold font indicates the best score in a given column for each model group. An asterisk indicates that a score is significantly better ($p < 0.05$) than the first-order version of that model. All significance tests were implemented using one-tailed non-parametric bootstrap resampling using 10,000 iterations.

model	creation time	matrix size
WA (1)	–	75 MB
WA (2)	33 seconds	1.8 GB
WA (3)	4.5 minutes	9.7 GB
WA (4)	8.6 minutes	19 GB
PR (1)	–	41 GB
PR (2)	45.6 hours	41 GB
PR (3)	45.6 hours	41 GB

Table 2: Runtime and memory requirements for creation of matrices, for the original PageRank and our cautious random walk algorithms. Creation of both higher-order WA and PR models is trivially parallelizable, and we report runtimes on a 2.7 GHz processor with parallel execution on 10 cores.

based NNLM (DN$_{\text{O}}$) is the best higher-order model. This is an encouraging result for DN$_{\text{O}}$, considering its simplicity. In general, NNLMs perform better than alignment models for answer reranking, which is not surprising considering the difference in size of the training datasets. To our knowledge, this is the first time this type of analysis has been performed for QA.

(v) Both PR$_{\text{U}}$(1-2) and PR$_{\text{P}}$(1-2) perform better than their first-order variants and better than our

#	Model/Features	P@1	P@1	MRR	MRR
			Impr.		Impr.
	Baselines				
1	Random	14.29		26.12	
2	CR	19.57	–	43.15	–
3	Jansen et al. (2014)	26.57	–	49.31	–
	Words				
4	CR + WA(1) + WN(1)	30.85	+58%	52.24	
5	CR + WA(1-2) + WN(1-2)	31.85*	+63%	52.86*	+23%
6	CR + WA(1-3) + WN(1-3)	**32.09***	**+64%**	**52.97***	**+23%**
7	CR + WA(1-4) + WN(1-4)	31.69	+62%	52.75	+22%
	Dependencies				
8	CR + DA(1) + DN$_\text{O}$(1) + DN$_\text{L-G}$(1)	31.49	+61%	52.93	+23%
9	CR + DA(1-2) + DN$_\text{O}$(1-2) + DN$_\text{L-G}$(1-2)	**32.85***	**+68%**	**53.73***	**+25%**
10	CR + DA(1-3) + DN$_\text{O}$(1-3) + DN$_\text{L-G}$(1-3)	32.77*	+67%	53.71*	+24%
	Words and Dependencies				
11	CR + WA(1) + WN(1) + DA(1) + DN$_\text{O}$(1) + DN$_\text{L-G}$(1)	31.85	+63%	53.03	+23%
12	CR + WA(1-2) + WN(1-2) + DA(1-2) + DN$_\text{O}$(1-2) + DN$_\text{L-G}$(1-2)	32.89†	+68%	53.69†	+24%
13	CR + WA(1-3) + WN(1-3) + DA(1-3) + DN$_\text{O}$(1-3) + DN$_\text{L-G}$(1-3)	**33.01†**	**+69%**	**53.96†**	**+25%**

Table 3: Performance on the Yahoo! Answers dataset for word, dependency, and models that combine the word and dependency representations. Bold font indicates the best score in a given column for each model group. An asterisk indicates that a score is significantly better ($p < 0.05$) than the first-order version of that model group (1), and † indicates that a score approaches significance ($p < 0.10$).

Model	P@1	P@1	MRR	MRR
		Delta		Delta
CR + WA(1)				
all features	27.33	–	49.20	–
− $P(Q\|A)$	25.69	-6%	48.35	-2%
− max JSD	27.33	0%	49.10	0%
− min JSD	23.57	-14%	46.64	-5%
− composite JSD	27.17	-1%	49.02	0%
− average JSD	25.41	-7%	47.70	-3%
CR + WN(1)				
all features	30.69	–	52.02	–
− max cosine sim.	29.65	-3%	51.49	-1%
− min cosine sim.	29.69	-3%	51.43	-1%
− composite cosine sim.	27.01	-12%	49.13	-6%
− average cosine sim.	26.49	-14%	49.04	-6%

Table 4: Ablation experiments for first-order word models. Each row removes a single feature from the corresponding complete model ("all features").

word-based second-order models (WA and WN). As expected, the personalized PR algorithm performs better than PR$_\text{U}$. However, their performance drops for the third-order models (rows 26 and 29), whereas our models continue to improve in their third-order configuration. All our third-order models perform better than all third-order PR models. This is caused by our "cautious" traversal strategy that considers only the closest neighbors during random walks, which controls for semantic drift better than the PR methods. Furthermore, the resources required for PR are considerably larger than the ones needed to implement our methods. Table 2 summarizes the requirements of the PR algorithm compared to our closest method (WA). As the table shows, PR has a runtime that is two orders of magnitude larger than WA's, and requires four times as much memory to store the generated higher-order matrices.

Combining NNLMs and alignment models

We explore combinations of NNLMs and alignment models in Table 3, which lists results when higher-order NNLMs and alignment models are combined for a given representation type. Each line in the table corresponds to a single answer reranking model, which incorporates features from multiple LS models. These experiments reinforce our previous observations: higher-order models perform better than their first-order variants regardless of representation type, but performance increases only for relatively low orders, e.g., 3 orders for words and words combined with dependencies, or 2 orders for dependencies.

Importantly, the table shows that the combination of NNLMs and alignment models across representation types is beneficial. For example, the best first-order combined model (row 11 in Table 3) performs similarly to the best *higher-order* individual LS model (row 18 in Table 1). The best combined higher-order model (row 13 in Table 3) has a P@1 score of 33.01, approximately 1.2% higher than the best individual LS model (row 18 in Table 1). To our knowledge, this is the highest performance reported on this Yahoo! Answers corpus, surpassing the best model of Jansen et. al (2014), which incorporates both shallow and deep discourse information, and achieves 32.9 P@1.

Feature ablation experiments

Although the main focus of this work is on higher-order LS models, Table 1 shows that even our first-order models perform better than the previous state of the art. This is caused by the novel features proposed over alignment models and NNLMs. To understand the contribution of each of these features, we performed an ablation experiment, summarized in Table 4, for two models: one alignment model (WA), and one NNLM (WN). This analysis indicates that many of the features proposed are important. For example, two of the novel JSD features for WA have a higher contribution to overall QA performance than the alignment probability ($P(Q|A)$) proposed in previous work (Surdeanu et al., 2011). For WN, the two new features proposed here (maximum and minimum cosine similarity between embedding vectors) also have a positive contribution to overall performance, but less than the other two features proposed in previous work (Yih et al., 2013; Jansen et al., 2014).

6.5 Discussion

One important observation yielded by the previous empirical analysis is that higher-order models perform well for sparse first-order variants (e.g., DA), but not for first-order models that rely on already dense association graphs (e.g., WN). We suspect that in densely populated graphs, modeling context becomes critical for successful higher-order models. Returning to our example from §1, a densely populated graph may already have the associations *breakfast – pancakes* and *pancakes – hashbrowns* that allow it to identify restaurants with favorable breakfast foods. Exploring higher-order associations in this situation is only beneficial if context is carefully maintained, otherwise an answer reranking system may erroneously select answer candidates with different contexts due to accumulated semantic drift (e.g., answers that discuss films, using associations from texts reviewing *Breakfast at Tiffany's*). Incorporating word-sense disambiguation or topic modeling into alignment or NNLM models may begin to address these contextual issues by preferentially associating terms within a given topic (such as *restaurants* or *films*), ultimately reducing semantic drift, and extending these higher-order methods beyond a few hops in the graph. Our analysis suggests that the extra sparsity that contextually-dependent representations introduce may make those models even more amenable to the higher-order methods discussed here.

More generally, we believe that graph-based inference provides a robust but approximate middle-ground to inference for QA. Where inference using first-order logic offers provably-correct answers but is relatively brittle (see Ch. 1 in MacCartney (2009)), LS methods offer robustness, but have lacked explanatory power and the ability to connect knowledge into short inference chains. Here we have demonstrated that higher-order methods capitalize on indirect evidence gathered by connecting multiple direct associations, and in doing so significantly increase performance over LS approaches that use direct evidence alone. By incorporating contextual information and making use of the short inference chains generated by traversing these graphical models, we hypothesize that graph-based systems will soon be able to construct simple justifications for their answer selection (e.g., *pancakes are a breakfast food, and hashbrowns go well with pancakes*). We hope it will soon be possible to fill this gap in inference methods for QA with higher-order LS models for robust but approximate inference.

7 Conclusions

We introduce a higher-order formalism that allows lexical semantic models to capitalize on both direct and indirect evidence. We demonstrate that many lexical semantic models, including monolingual alignment and neural network language models, working over surface or syntactic representations, can be trivially adapted to this higher-order formalism. Using a corpus of thousands of non-factoid *how* questions, we experimentally demonstrate that higher-order methods perform better than their first-order variants for most lexical semantic models investigated, with statistically-significant relative gains of up to 13% over the corresponding first order models.

Acknowledgements

We thank the Allen Institute for Artificial Intelligence for funding this work. We would also like to thank the three anonymous reviewers for their helpful comments and suggestions.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*.

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*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*.

Adam Berger, Rich Caruana, David Cohn, Dayne Freytag, and Vibhu Mittal. 2000. Bridging the lexical chasm: Statistical approaches to answer finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research & Development on Information Retrieval*.

Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and Shashank Sudarshan. 2002. Keyword searching and browsing in databases using BANKS. In *Proceedings of 18th International Conference on Data Engineering (ICDE)*.

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.

Soumen Chakrabarti and Alekh Agarwal. 2006. Learning parameters in entity relationship graphs from ranking preferences. In *Proceedings of 10th European Conference on Principle and Practice of Knowledge Discovery in Databases (PKDD)*.

Soumen Chakrabarti. 2007. Dynamic personalized PageRank in entity-relation graphs. In *Proceedings of the 16th International World Wide Web Conference (WWW)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

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)*.

Oren Etzioni. 2011. Search needs a shake-up. *Nature*, 476(7358):25–26.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL-1998)*.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tara McIntosh. 2010. *Reducing Semantic Drift in Biomedical Lexicon Bootstrapping*. Ph.D. thesis, University of Sydney.

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 (ICLR)*.

Einat Minkov, William W. Cohen, and Andrew Y. Ng. 2006. Contextual search and name disambiguation in email using graphs. In *Proceedings of SIGIR*.

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*, pages 684–691.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Radu Soricut and Eric Brill. 2006. Automatic question answering using the web: Beyond the factoid. *Journal of Information Retrieval - Special Issue on Web Information Retrieval*, 9(2):191–206.

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*, 2:219–230.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Proceedings of the 6th International Conference on Data Mining (ICDM)*.

Xin-Jing Wang, Xudong Tu, Dan Feng, and Lei Zhang. 2009. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the Annual ACM SIGIR Conference*.

Max Whitney and Anoop Sarkar. 2012. Bootstrapping via graph propagation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Semi-Markov phrase-based monolingual alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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 (ACL)*.

Benat Zapirain, Eneko Agirre, Lluis Marquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3).