Neural Information Retrieval: A Literature Review

Md Mustafizur Rahman
University of Texas at Austin
15 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Henna Kim
University of Texas at Austin
9 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Neural Information Retrieval: A Literature Review

Ye Zhang*, Md Mustafizur Rahman, Alex Braylan,
Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert,
Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu,
Byron C. Wallace, Matthew Lease†
*Department of Computer Science, University of Texas at Austin, yezhang@utexas.edu
†School of Information, University of Texas at Austin, ml@utexas.edu

Abstract

A recent "third wave" of Neural Network (NN) approaches now delivers state-of-the-art performance in many machine learning tasks, spanning speech recognition, computer vision, and natural language processing. Because these modern NNs often comprise multiple interconnected layers, this new NN research is often referred to as *deep learning*. Stemming from this tide of NN work, a number of researchers have recently begun to investigate NN approaches to Information Retrieval (IR). While deep NNs have yet to achieve the same level of success in IR as seen in other areas, the recent surge of interest and work in NNs for IR suggest that this state of affairs may be quickly changing. In this work, we survey the current landscape of *Neural IR* research, paying special attention to the use of learned representations of queries and documents (i.e., neural embeddings). We highlight the successes of neural IR thus far, catalog obstacles to its wider adoption, and suggest potentially promising directions for future research.

Keywords: convolutional neural network (CNN), deep learning, distributed representations, neural network (NN), recurrent neural network (RNN), search engine, word embedding, `word2vec`

1 Introduction

We are in the midst of a tremendous resurgence of interest and renaissance in research on artificial neural network (NN) models for machine learning, now commonly referred to as *deep learning*[1]. While many valuable introductory readings, tutorials, and surveys already exist for deep learning at large, we are not familiar with any existing literature review surveying NN approaches to Information Retrieval (IR). Given the great recent rise of interest in such *Neural IR* from both researchers (Gao, 2015; Li and Lu, 2016) and practitioners (Metz, 2016; Ordentlich et al., 2016) alike, we believe that such a literature review is now timely. IR researchers interested in getting started with Neural IR currently must identify and compile many scattered works. Unifying these into a coherent resource would provide a single point of reference for those interested in learning about these emerging approaches, as well as provide a valuable reference compendium for more experienced Neural IR researchers.

To address this need, this literature review surveys recent work in Neural IR. Our survey is intended for IR researchers already familiar with fundamental IR concepts and so requiring few definitions or explanations of these. Those less familiar with IR may wish to consult existing reference materials (Croft et al., 2009; Manning et al., 2008) for unfamiliar terms or concepts. However, we anticipate many readers will have relatively less familiarity and experience with NNs and deep learning. Consequently, we briefly introduce key terms, definitions, and concepts in Sections 3 and 4. Because many exemplary introductory resources on deep learning already exist, we strive to avoid duplicating this material and instead encourage readers to directly consult these resources for additional background. For general introductions to deep learning, see Goodfellow et al. (2016); LeCun et al. (2015); Deng (2014); Yu and Deng (2011); Schmidhuber (2015); Arel et al. (2010), and Bengio (2009). See Broder et al. (2016) for a recent panel discussion on deep learning. For introductions to deep learning approaches to other domains, see Hinton et al. (2012) for automatic speech recognition (ASR), Goldberg (2015) for natural language

[1]While not all NNs are 'deep', and not all 'deep' models are neural, the terms are often conflated in practice.

processing (NLP), and Wu et al. (2016) for machine translation (MT). Regarding NN approaches to IR, informative talks and tutorials have been presented by Gao (2015); Li and Lu (2016), and Li (2016a,b). Many other useful tutorials and talks can be found online for general deep learning and specific domains. A variety of resources for deep learning, including links to popular open-source software, can be found online at `http://deeplearning.net`.

In terms of scope, we restrict this survey to textual IR. For NN approaches to content-based image retrieval, see Wan et al. (2014). Similarly, we exclude work on NN approaches to acoustic or multi-modal IR, such as mixing text with imagery (see Ma et al. (2015b, 2016)). We also intentionally focus on the current "third wave" revival of NN research, excluding earlier work. Finally, we largely follow the traditional divide between IR and NLP, including search-related IR research and excluding syntactic and semantic NLP work. However, this division is perhaps most difficult to enforce in regard to question answering (QA) and community QA (CQA) tasks, which perhaps represent the greatest cross-over between NLP and IR fields (e.g., see Dumais et al. (2002)). Recent years have seen QA research being more commonly published in NLP venues and often more focused on semantic understanding of short texts rather than search over large collections. As a pragmatic and imperfect compromise, we largely make the divide by publication venue, including those works appearing at IR venues and excluding those from NLP venues. For readers interested in further reading about deep QA, see (Bordes et al., 2014; Kumar et al., 2015; Gao, 2015; Goldberg, 2015).

Finally, regarding nomenclature, our use of *Neural IR*, referring to machine learning research on *artificial* NNs and deep learning, should not be confused with cognitive science research studying *actual* neural representations of relevance in the human brain (see (Moshfeghi et al., 2016)). In addition, note that the modern appellation of *deep learning* for the current "third wave" of NN research owes to these approaches using a large number of *hidden layers* in their NN architectures. For this literature review, we have chosen the term *neural* (rather than *deep*) because: i) most current work on textual NNs in NLP and IR is often actually quite shallow in the number of layers used (typically only a few hidden layers and often only one, though some notable recent exceptions exist, such as Conneau et al. (2016)); and ii) use of *neural* more clearly connects the current wave of *deep learning* to the long history of NN research.

The remainder of this literature review is organized as follows. To illustrate the rough evolution of Neural IR research over time, we begin by providing a concise history of Neural IR in Section 2. Following this, in Section 3 we introduce the concept of *word embeddings* and survey approaches that extend traditional IR models to integrate such embeddings, often using off-the-shelf `word2vec` source code or trained embeddings. In contrast, Section 5 surveys approaches which more fully embrace NN modeling. Such models treat word embeddings as the first layer in a network, to be learned along with all model parameters. This is an exciting direction, as researchers are broadly working toward developing *end-to-end* NN architectures for IR, which may depart significantly from traditional IR models. However, assuming a readership familiar with IR but possibly less versed in neural network (NN) concepts, Section 4 first briefly introduces several such concepts which underlie work in Section 5. Section 6 provides discussion, and we conclude in Section 7. Table 3 lists datasets used in Neural IR studies to date and the studies reporting on each. Tables 4 and 5, respectively, list source code and data shared from published studies.

2 A Brief History of Neural IR

Introductory resources on deep learning cited in Section 1 (see LeCun et al. (2015) and Goodfellow et al. (2016)) explain how the "third wave" of interest in neural network approaches arose. Key factors include increased availability of "big data", more powerful computing resources, and better NN models and parameter estimation techniques. While early use in language modeling for ASR and MT might be loosely related to language modeling for IR (Ponte and Croft, 1998), state-of-the-art performance provided by neural language models trained on vast data could not be readily applied to the more typical sparse data setting of training document-specific language models in IR. Similarly, neural approaches in computer vision to learn higher-level representations (i.e., *features*) from low-level pixels were not readily transferable to text-based research on words.

However, a rapid spread and adoption in NLP was sparked when Mikolov and Dean (2013) and Mikolov et al. (2013) proposed `word2vec`, a relatively simple model and estimation procedure for *word embeddings* (also known as *distributed term representations*), coupled with sharing both source code and trained embeddings (see Tables 4 and 5). This availability of `word2vec` code and existing embeddings has provided one of the key avenues for early work on neural IR, especially for extending traditional IR models to integrate word embeddings. Given the significance and prevalence of such research, we review this body of work first in Section 3. We also believe this organization is helpful to conceptually differentiate approaches that integrate word embeddings into traditional IR models vs. approaches directly incorporating word embeddings within NN models, reflecting a more significant shift toward pursuing *end-to-end* NN architectures in IR (Section 5).

Salakhutdinov and Hinton (2009) proposed the first "third wave" Neural IR work we are aware of, employing a deep *auto-encoder* architecture (Section 4) for semantic modeling for related document search (Section 5.6). They did not have relevance judgments for evaluation, so instead used document corpora with category labels and assumed relevance if a query and document had matching category labels.

Ad-hoc search was addressed more directly in 2013. Huang et al. (2013) proposed a Deep Structured Semantic Model (DSSM)[2], which has generated a variety of follow-on work (e.g., (Gao et al., 2014; Shen et al., 2014a,b; Mitra, 2015; Mitra and Craswell, 2015; Song et al., 2016)). DSSM's feed-forward NN learned low-dimensional vector representations (embeddings) of queries and documents, meant to capture the latent semantics in the texts. Because vocabulary size is typically quite large in real-world Web search, the authors were concerned that when using term vectors as input to the NN, the NN input layer size would become unmanageable for model training and inference. To address this, a "word hashing" dimensionality reduction technique was proposed to represent words based on their character trigrams (see Section 6.2). Because such hashing also maps orthographic variants of the same word to nearby points in the hashing space, such hashing was also useful in handling out-of-vocabulary (OOV) query terms (how to best address OOV query terms not found in trained word embeddings remains an open question). While DSSM was applied to ad-hoc websearch, the challenge of effectively supporting full-text search led the authors to instead index document titles only. The challenge to match ad-hoc search effectiveness of traditional IR models with full document indexing using Neural IR approaches has been oft-discussed up to the present day (e.g., Cohen et al. (2016); Guo et al. (2016)).

Also in 2013, Clinchant and Perronnin (2013) anticipated work on *word embedding* approaches to IR which would later stem from `word2vec`. In this case, classic Latent Semantic Indexing (LSI) (Deerwester et al., 1990) was used to induce word embeddings, which were transformed into fixed-length Fisher Vectors (FVs) via Jaakkola et al. (1999)'s Fisher Kernel (FK) framework, and then compared via cosine similarity for document ranking. Results for ad-hoc search on three IR test collections were reported, though their proposed approach was outperformed by traditional Divergence From Randomness (DFR) (Amati and Van Rijsbergen, 2002) ranking. That same year, Lu and Li (2013) also proposed *DeepMatch*. Similar to Huang et al. (2013), the authors focused on short-text matching, testing their approach on two datasets: community question answering (CQA) (matching questions with answers) and a Twitter-like micro-blog task (matching tweets with comments).

In 2014, in addition to the DSSM follow-on works cited above (Shen et al., 2014a,b; Gao et al., 2014), a variety of further work began to appear. Sordoni et al. (2014) investigated query expansion, evaluating ad-hoc search on TREC collections. Le and Mikolov (2014) proposed their `ParagraphVector` method for composing word embeddings to induce semantic representations over longer textual units (see Section 6.3). Their evaluation included an unusual IR task of detecting which of three results snippets did not belong for a given query (see *Outlier Detection* in Section 3.4). Li et al. (2014) used embeddings as input to a feed-forward deep NN to exploit a user's previous queries in order to improve document ranking. The first two neural IR papers also appeared at ACM SIGIR (Gupta et al., 2014; Zhang et al., 2014). Gupta et al. (2014) proposed an auto-encoder approach to mixed-script query expansion, considering transliterated search queries and learning a character-level "topic" joint distribution over features

[2]`https://www.microsoft.com/en-us/research/project/dssm`

of both scripts. Zhang et al. (2014) considered the task of local text reuse, with three annotators labeling whether or not a given passage represents reuse. As in Clinchant and Perronnin (2013), they adopted the Fisher Kernel (FK) to reduce variable-size word embedding concatenations to fixed length, and they further tested different hashing methods to reduce dimensionality.

By 2015, work on neural IR had grown beyond what can be concisely described here. We saw `word2vec` enter wider adoption in IR research (e.g., Ganguly et al. (2015); Grbovic et al. (2015a); Kenter and de Rijke (2015); Zheng and Callan (2015); Zuccon et al. (2015)), as well as a flourishing of neural IR work appearing at SIGIR (Ganguly et al., 2015; Grbovic et al., 2015b; Mitra, 2015; Severyn and Moschitti, 2015; Vulic and Moens, 2015; Zheng and Callan, 2015), spanning ad-hoc search (Ganguly et al., 2015; Vulic and Moens, 2015; Zheng and Callan, 2015; Zuccon et al., 2015), QA sentence selection and Twitter reranking (Kenter and de Rijke, 2015), cross-lingual IR (Vulic and Moens, 2015), paraphrase detection (Kenter and de Rijke, 2015), query completion (Mitra, 2015), query suggestion (Sordoni et al., 2015), and sponsored search (Grbovic et al., 2015a,b). We also saw in 2015 the first workshop on Neural IR[3].

As of 2016, work on neural IR continues to accelerate in quantity of work, sophistication of methods, and practical effectiveness (e.g., Guo et al. (2016)). SIGIR also featured its first workshop on the subject[4], as well as a tutorial by Li and Lu (2016). To provide as current of coverage as possible in this literature review, we include articles appearing up through ACM ICTIR 2016 and CIKM 2016 conferences.

3 Word Embedding Approaches to IR

This section surveys recent studies extending traditional IR models to incorporate word embeddings. We distinguish these studies from approaches directly incorporating word embeddings within NN models (see Section 5), reflecting a more significant shift toward pursuing *end-to-end* NN architectures in IR. We begin by presenting background on word embeddings in Section 3.1. Following this, we organize studies surveyed by IR tasks (see Table 1). For additional discussion and comparison of methods, see Section 6.

3.1 Background

Terms have traditionally been encoded as discrete symbols which cannot be directly compared to one another (unless we consider edit distance at the character-level). Conceptually, this is equivalent to a *one-hot encoding* in which each term is represented as a sparse vector with dimensionality equal to the vocabulary size (each dimension corresponding to a unique word). To one-hot encode a given term t, we create a $\mathbf{0}$ vector and set the index corresponding to t to 1. In such an encoding, all terms are orthogonal and equidistant to one another, and hence there is no easy way to recognize similar terms in the vector space. This has led to the infamous *vocabulary mismatch* problem in which an IR system must recognize when distinct but related terms occur in query and document to perform effective matching.

In contrast with such a one-hot representation, *word embeddings* (also known as *distributed term representations*) encode each symbol as a low-dimensional (say, hundreds of dimensions), continuous, dense vector. Following oft-cited Firth (1957)'s adage that "You shall know a word by the company it keeps", by capturing the extent to which words occur in similar contexts, word embeddings are able to encode semantic and syntactic similarity insofar as the embeddings for similar words will be nearby one another in vector space. This allows one to perform simple algebraic operations that reflect the word's meaning, e.g., *wv*("Madrid") - *wv*("Spain") + *wv*("France") is expected to be close (e.g., with respect to Euclidean distance) to *wv*("Paris").

Word embeddings are typically induced using large unlabeled corpora in an unsupervised way. The most popular method and software for learning word embeddings today, `word2vec`, was proposed by Mikolov and Dean (2013) and Mikolov et al. (2013). `word2vec` is a model that uses a three-layer NN with one hidden layer to learn word representations. Two variants were proposed: *skip-gram* predicts the surrounding context given the current word, while *continuous bag-of-words* (CBOW) predicts the

[3]`http://research.microsoft.com/en-us/um/beijing/events/DL-WSDM-2015/`

[4]`https://www.microsoft.com/en-us/research/event/neuir2016/`

current word given its context. The authors trained `word2vec` on a large news corpus, releasing both software and learned embeddings online (see Tables 4 and 5).

As an alternative to `word2vec`, Pennington et al. (2014) proposed Global Vectors for Word Representation (GloVe), also sharing source code and learned embeddings online (see Tables 4 and 5). In contrast to `word2vec`'s language modeling based on contextual windows, GloVe induces a log-bilinear regression model based on global word-word co-occurrence counts. The practical efficacy of using one embedding vs. another has been considered in different IR studies (see Kenter and de Rijke (2015)), and it is not clear that any method or embedding set performs best in all situations. Models that jointly exploit multiple sets of embeddings may therefore be appealing (Zhang et al., 2016b; Neelakantan et al., 2014). See further discussion of word embeddings in Section 6.1.

There is a long history of distributional approaches that is sometimes overlooked given current interest in `word2vec`. As briefly mentioned in Section 2, Clinchant and Perronnin (2013) use classic Latent Semantic Indexing (LSI) (Deerwester et al., 1990) to induce word embeddings, as well as Jagarlamudi et al. (2011)'s Yule association measure. The authors also note Collobert and Weston (2008)'s embedding. In distributional semantics, see prior work by Kiela and Clark (2013), which Lioma et al. (2015) build upon in an IR context to infer whether query terms should be processed separately or as a single phrase for matching. In relation to prior contextual window approaches, such as the hyperspace analogue to language (HAL) (Lund and Burgess, 1996) and probabilistic HAL (Azzopardi et al., 2005) (see also (Bruza and Song, 2002)), Zuccon et al. (2015) provide a succinct but valuable comparison to `word2vec`: co-occurrences in HAL within a window centered on a target term are accumulated, while `word2vec` skip-gram fits the representation of a target term to predict its lexical context (and vice-versa for CBOW). Zuccon et al. (2015) conclude that as of 2015, "it is not clear yet whether these neural inspired models are generally better than traditional distributional semantic methods."

Task	**Studies**
Ad-hoc Retrieval	ALMasri et al. (2016), Amer et al. (2016), BWESG (Vulic and Moens (2015)), Clinchant and Perronnin (2013), Diaz et al. (2016), GLM (Ganguly et al. (2015)), Mitra et al. (2016), Nalisnick et al. (2016), NLTM (Zuccon et al. (2015)),Rekabsaz et al. (2016), Roy et al. (2016), Zamani and Croft (2016a), Zamani and Croft (2016b), Zheng and Callan (2015)
Bug Localization	Ye et al. (2016)
Contextual Suggestion	Manotumruksa et al. (2016)
Cross-lingual IR	BWESG (Vulic and Moens (2015))
Detecting Text Reuse	Zhang et al. (2014)
Domain-specific Semantic Similarity	De Vine et al. (2014)
Community Question Answering	Zhou et al. (2015)
Short Text Similarity	Kenter and de Rijke (2015)
Outlier Detection	`ParagraphVector` (Le and Mikolov (2014))
Sponsored Search	Grbovic et al. (2015b), (Grbovic et al., 2015a)

Table 1: IR tasks solved by embedding approaches.

3.2 Ad-hoc Retrieval

Ad-hoc retrieval refers to the initial search performed by a user: a single query, with no further interaction or feedback, on the basis of which an IR system strives to return an accurate document ranking.

A series of international shared task evaluations for search systems have been established over the years, providing a wealth of reusable search collections, topics, and relevance judgments for training and evaluating IR systems (see Sanderson (2010)):

- the Text REtreival Conference (TREC)[5] (Voorhees and Harman, 2005), begun in 1992
- the NII Testbeds and Community for Information access Research (NTCIR)[6], begun in 1999
- the Conference and Labs of the Evaluation Forum (CLEF), formerly known as the Cross-Language Evaluation Forum[7] (Braschler and Peters, 2004), begun in 2000
- the Initiative for the Evaluation of XML Retrieval (INEX)[8] (Lalmas and Tombros, 2007), begun in 2002,
- the Forum for Information Retrieval Evaluation (FIRE)[9], begun in 2008

A number of open-source toolkits have also been made available to enable and ease repeatable IR search experiments: Galago[10], Indri[11], Lemur[12], Lucene[13], and Terrier[14].

As briefly mentioned in Section 2, **Clinchant and Perronnin (2013)** present recent pre-`word2vec` work using word embeddings in IR. The authors articulate their primary contribution as showing that a document can be represented as a bag-of-embedded-words (BoEW). Latent Semantic Indexing (LSI) (Deerwester et al., 1990) is used to induce word embeddings, which are then transformed into fixed-length Fisher Vectors (FVs) via Jaakkola et al. (1999)'s Fisher Kernel (FK) framework. The FVs are then compared via cosine similarity for document ranking. Experiments on ad-hoc search using Lemur are reported for three collections: TREC ROBUST04, TREC Disks 1&2, and English CLEF 2003 Ad-hoc. While results show improvement over standard LSI on all three collections, standard TF-IDF performs slightly better on two of the three collections, and Divergence From Randomness (DFR) (Amati and Van Rijsbergen, 2002) performs far better on all collections. The authors note, "These results are not surprising as it has been shown experimentally in many studies that latent-based approaches such as LSI are generally outperformed by state-of-the-art IR models in Ad-Hoc tasks." Document clustering results are also reported. The authors note that their approach is independent of any particular embedding technique, and briefly report on using a word embedding based on Jagarlamudi et al. (2011)'s Yule association measure instead of LSI. The authors propose to consider further embeddings in future work, such as Collobert and Weston (2008)'s embedding. Finally, the authors note that another potential advantage of their proposed framework is the ability to seamlessly deal with multilingual documents (e.g., see Vulic and Moens (2015)).

Ganguly et al. (2015) propose a Generalized Language Model (GLM) for integrating word embeddings with query-likelihood language modeling. Semantic similarity between query and document/collection terms is measured by cosine similarity between word embeddings induced via `word2vec` CBOW. The authors frame their approach in the context of classic *global* vs. *local* term similarity, with word embeddings trained without reference to queries representing a global approach akin to the Latent Dirichlet Allocation (LDA) of Wei and Croft (2006). Like Rekabsaz et al. (2016) and Zuccon et al. (2015), the authors build on Berger and Lafferty (1999)'s "noisy channel" translation model. The noisy channel may transform a term t' observed in a document into a term t observed in a query, either by document sampling or collection sampling. For document sampling, they take

[5]`http://trec.nist.gov`
[6]`http://research.nii.ac.jp/ntcir/index-en.html`
[7]`http://www.clef-initiative.eu`
[8]`http://inex.mmci.uni-saarland.de`
[9]`http://fire.irsi.res.in`
[10]`http://www.lemurproject.org/galago.php`
[11]`http://www.lemurproject.org/indri.php`
[12]`http://www.lemurproject.org/lemur.php`
[13]`http://lucene.apache.org`
[14]`http://terrier.org`

$P(t, t'|d) = P(t|t', d)P(t'|d)$, where $P(t|t', d)$ is computed based on cosine similarity between embeddings of t and t'. For a single document, $P(t, t'|d)$ considers all document terms, but for transformation via the collection C, $P(t, t'|C)$ considers only a small neighborhood of terms around query term t to reduce computation. Smoothing of mixture model components resembles classic cluster-based LM smoothing of Liu and Croft (2004). It is unclear how out-of-vocabulary (OOV) query terms are handled. Ad-hoc search results reported for TREC 6-8 and Robust using Lucene show improvement over both unigram query-likelihood and LDA. However, parameters appear to be tuned on testing collections, and LDA results are much lower than Wei and Croft (2006)'s, which the authors hypothesize is due to training LDA on the entire collection rather than collection subsets. The authors do not compare their global approach vs. local pseudo-relevance feedback (PRF) (Lavrenko and Croft, 2001), which prior has shown to outperform LDA (Yi and Allan, 2009), and while typically a query-time technique, can be approximated for greater efficiency (Cartright et al., 2010).

Zuccon et al. (2015) propose a Neural Language Translation Model (NLTM), also integrating word embeddings into Berger and Lafferty (1999)'s classic translation model approach to query-likelihood IR. They estimate the translation probability between terms as the cosine similarity of the two terms divided by the sum of the cosine similarities between the translating term and all of the terms in the vocabulary. However, note that use of cosine similarity as a proxy for a probability assumes word embeddings have purely positive values. Previous state-of-the-art translation models use mutual information (MI) embeddings to estimate translation probability. Experiments evaluating the NLTM approach on ad-hoc search are reported on the TREC datasets AP87-88, WSJ87-92, DOTGOV, and MedTrack. Results indicate that the NLTM approach provides moderate improvements over the MI and classic TM systems, based on modest improvements to a large number of topics, rather than large differences on a few topics. Sensitivity analysis of the various model hyper-parameters for inducing word embeddings shows that manipulations of embedding dimensionality, context window size, and model objective (CBOW vs skip-gram) have no consistent impact upon NLTM's performance vs. baselines. Regarding choice of training corpus for learning embeddings vs. search effectiveness, although effectiveness typically appears highest when embeddings are estimated using the same collection in which search is to be performed, the differences are not statistically significant. Source code and learned embeddings are shared online (see Tables 4 and 5).

Similar to *RegressionRank* (Lease et al., 2009), **Zheng and Callan (2015)** propose DeepTR to learn effective query term weights through supervision. However, whereas Lease et al. (2009) engineer features, Zheng and Callan (2015) automatically construct feature vectors using `word2vec` CBOW word embeddings. In particular, each query term's feature vector is constructed by simple subtraction of the average embedding vectors of query terms from the given term's embedding vector. L1 LASSO regression is used to predict a target weight for each query term given its feature vector. The *recall weight* (Zhao and Callan, 2010) of each term, learned from relevance judgments, is used as the target term weight for regression. In-collection training of target weights is assumed (i.e., it is assumed that relevance judgments are available for the same collection which is to be searched), using 5-fold cross-validation, with 80% of queries (and their relevance judgments) used for training and 20% used for testing. Experiments on ad-hoc search are performed with Indri on 4 TREC test collections: Robust04, WT10t, GOV2, and ClueWeb09-Cat-B, comparing against 3 baselines: BM25, unigram language modeling (LM), and Metzler and Croft (2005)'s sequential dependency model. Results are reported only for verbose TREC `Description` queries, rather than keyword `title` queries (and analysis by the authors shows that DeepTR performs much better on queries of 4 or more words). DeepTR is shown to achieve statistically significant improvements over baselines, showing that term embeddings can be effectively exploited to improve ad-hoc search accuracy without requiring end-to-end neural network modeling. Regarding embedding dimensionality, Zheng and Callan (2015) find that 100 dimensions work best for estimating term weights, better than 300 and 500. Training on external corpora performs best, though no single corpus is consistently best, despite widely varying size of training data. No psuedo-relevance feedback (Lavrenko and Croft, 2001) experiments are reported, though the authors suggest that embeddings may help identify expansion terms.

A pair of studies by **Zamani and Croft (Zamani and Croft, 2016a,b)** investigate related word embedding approaches. Similar to Zheng and Callan (2015), **Zamani and Croft (2016b)** develop a method for query embedding based on the embedding vectors of its individual words. However, whereas Zheng and Callan (2015)'s simple average of vectors is heuristic, Zamani and Croft (2016b) develop a theoretical framework. Their intuitive idea is that a good query vector should yield a probability distribution $P(w|q) = \delta(w, q)/Z$ induced over the vocabulary terms similar to the query language model probability distribution (as measured by the KL Divergence). Here, w is the embedding vector of the word, q is the query embedding vector, $\delta(w, q)$ is the similarity between the two embedding vectors, and Z is a normalization constant. For similarity, they use the softmax and sigmoid transformations of cosine similarity. The authors show that the common heuristic of averaging individual query term vectors to induce the overall query embedding is actually a special case of their proposed theoretical framework for the case when vector similarity is measured by softmax and the query language model is estimated by maximum likelihood. Experiments with ad-hoc search using Galago are reported for three TREC test collections: AP, Robust04, and GOV2, using keyword TREC `title` queries. Word embeddings are induced by GloVe (Pennington et al., 2014). Two different methods are used for estimating the query language model: via Pseudo Relevance Feedback (PRF) (Lavrenko and Croft, 2001), which they refer to as *pseudo query vector* (PQV), and via maximum-likelihood estimation (MLE). Two different methods are used for calculating similarity: Softmax and Sigmoid. Softmax is easy to use because of the closed form solution, but the sigmoid function consistently showed better performance, likely due to the flexibility provided by its two free parameters. Although PQV shows higher mean average precision, precision of top documents appears to suffer. Results also emphasize the importance of training data selection. Embeddings trained from the same domain as the documents being searched perform better than embeddings trained on a larger dataset from a different domain. Increasing the dimensionality of embedding vectors seems to improve retrieval, but it is unclear what would happen beyond the 300 dimensions considered. The authors also present query classification results (see Section 3.4). With regard to future work, because the current work assumes that embedding vectors for all query terms are available, one future direction could explore smoothing instead. The normalization term Z is also ignored by making a simplifying assumption that Z is the same across all queries, which does not hold in practice. Future research might seek to compute Z more efficiently or propose an alternative embedding distribution.

In their other study, **Zamani and Croft (2016a)** characterize their primary contribution vs. prior work as being their focus on applying embedding to queries rather than documents, estimating query language models which efficiently outperform embedding-based document language models (Ganguly et al., 2015). Moreover, whereas ALMasri et al. (2016) propose a heuristic method for query expansion VEXP based on similarity of word embeddings, Zamani and Croft (2016a) consider expansion of the whole query rather than term-by-term expansion. Also notable, while similarity of embedding vectors is often measured by cosine or Euclidean distance, the authors propose transforming the similarity values using sigmoid and show the empirical benefit of doing so. Two embedding-based query expansion (EQE) approaches are proposed. Similar to RM1 and RM2 in Lavrenko and Croft (2001)'s relevance models, Zamani and Croft (2016a) EQE1 assumes conditional independence between query terms, whereas EQE2 assumes the semantic similarity between two terms is independent of the query. The authors also propose an embedding-based relevance model (ERM). As in Zamani and Croft (2016b), ad-hoc search experiments with Galago are reported for TREC AP, Robust04, and GOV2 test collections using keyword TREC `title` queries, with word embeddings induced via GloVe (Pennington et al., 2014). Baselines include Ganguly et al. (2015)'s GLM and ALMasri et al. (2016)'s VEXP. The authors also consider an unsupervised variant baseline of Zheng and Callan (2015) based on the similarity of vocabulary term vectors and the average embedding vector of all query terms (AWE). PRF experiments compare RM1 and RM2 vs. ERM. Proposed methods tend to outperform baselines, and EQE1 tends to outperform EQE2. While the authors consider training GloVe on three external corpora, they find that "there is no significant differences between the values obtained by employing different corpora for learning the embedding vectors." The authors also present a sensitivity analysis of the sigmoid parameters. For future work, the authors propose modifying the learning process of embedding vectors to yield discriminative similarity values suited to many IR tasks. They also suggest further theoretical analysis of their use of sigmoid.

Nalisnick et al. (2016); Mitra et al. (2016) propose a Dual Embedding Space Model (DESM), writing that "a crucial detail often overlooked when using `word2vec` is that there are two different sets of vectors... IN and OUT embedding spaces [produced by `word2vec`]... By default, `word2vec` discards W_{OUT} at the end of training and outputs only W_{IN}..." In contrast, the authors retain both input and output embeddings. Query terms are mapped to the input space and document words to the output space. Documents are embedded by taking an average (weighted by document term frequency) over embedding vectors of document terms. Query-document relevance is computed by average cosine similarity between each query term and the document embedding. The authors induce word embeddings via `word2vec` CBOW only, though note that skip-gram embeddings could be used interchangeably. Embeddings learned on a Bing query log are compared to embeddings learned on a proprietary Web corpus. Results show "DESM to be a poor standalone ranking signal on a larger set of documents", so a re-ranking approach is proposed in which the collection is first pruned to all documents retrieved by an initial Bing search, and then re-ranked by DESM. Experiments with ad-hoc search are carried out on a proprietary Web collection using both explicit and implicit relevance judgments. In contrast with (Huang et al., 2013)'s DSSM, full Webpage texts are indexed instead of only page titles. DESM's IN and OUT embedding space combinations are compared to baseline retrieval by BM25 and latent semantic analysis (LSA) (Deerwester et al., 1990). Out-of-vocabulary (OOV) query terms are ignored for their DESM approach but retained for baselines. Re-ranking results show improvement over baselines, especially on the implicit feedback test set, with best performance when word embeddings are trained on queries and using IN-OUT embedding spaces in document ranking. The authors surmise that query-training performs better due to users tending to include only significant terms from their queries. For future work, the authors propose investigating using of IN and OUT embeddings in pseudo-relevance feedback and query expansion. They also propose investigating further ways to induce document embeddings and to seek a principled way to avoid the need for two-stage re-ranking. Learned word embeddings are shared online (see Table 5).

Vulic and Moens (2015) present a unified framework for monolingual and cross-lingual IR based on word embeddings. Section 3.4 discusses their approach to learning bilingual embeddings and their findings for cross-lingual search. We focus here on their monolingual model and results. Word-embeddings are learned via `word2vec`'s skip-gram model (the approach does not appear to be specific to skip-gram, though use of CBOW or GloVe embeddings is not discussed). Given term embeddings, query and document embeddings are induced by simply adding their constituent term vectors (we assume query and document vectors are normalized to avoid length bias, though this is not discussed). They also propose an extended approach in which the sum is weighted by each term's information-theoretic self-information, which akin to IDF is expected to signify more important terms. Documents are then ranked by cosine similarity between query and document embedding vectors. Experiments on ad-hoc search are performed on CLEF 2001-2003 test collections, reporting unigram language modeling and LDA (Wei and Croft, 2006) as baselines. The authors also report mixture models combining baselines and combining unigram modeling with their embedding approach, tuning the mixture weight. Queries used appear to be the concatenation of both `title` and `description` topic fields, so verbose but with key terms repeated. `word2vec` dimensions are varied from 100-800 in steps of 100, and window size is varied from 10-100 in steps of 10. Parameters appear to be tuned on testing data, suggesting upperbound performance achievable by each method (rather than expected performance in practice). The proposed embedding approach outperforms LDA, is outperformed by unigram modeling, and performs best in mixture with the unigram. Composing vectors weighted by self-information delivers small but consistent improvements over simple addition. For future work, the authors propose to investigate other vector composition models and use of embeddings in pseudo-relevance feedback.

Diaz et al. (2016) learn query-specific *local* word embeddings, rather than the typical practice of inducing *global* embeddings from a corpus without reference to user queries (see useful related discussion on local vs. global term similarity in (Ganguly et al., 2016; Zuccon et al., 2015)). This is the only work to date we are aware of that learns and exploits topic-specific word embeddings for ad-hoc search. The authors use the document retrieval scores from each query to learn a document relevance probability distribution over the document collection. They then sample a set of k documents according to this probability distribution, training word embeddings on this topically-biased document subset. The locally trained words embeddings are then used to derive an expansion language model for the query, which is interpolated with the original query language model in usual fashion to mitigate risk of *query drift*. Reported experiments use three TREC datasets: TREC1&2, Robust, and ClueWeb09-Cat-B. The authors train local word embedding on these three test collections, as well as on two external corpora: Gigaword and Wikipedia. Similar to Singhal and Pereira (1999), queries are run first on external documents to retrieve related documents for expansion, which are then used to improve the expansion of queries to be run on the test collection of interest. The proposed query expansion strategy outperforms a *global* baseline in which word embeddings are induced irrespective of queries. While these results are certainly encouraging, the computational cost of inducing word embeddings at query-time is expensive. Consequently, the authors suggest future work should address efficiency, perhaps by pre-computing some smaller number of topical embeddings at coarser granularity, and then simply selecting between these alternative topical embeddings at query-time.

As mentioned in discussion of Zamani and Croft (2016a), **ALMasri et al. (2016)** propose a heuristic method VEXP for term-by-term query expansion using embedding vectors. For each query term, it collects its several most similar terms in the embedding space and adds them to the query. Experiments with ad-hoc search use Indri on four CLEF medical test collections: Image2010-2012 (short documents and queries, text-only) and Case2011 (long documents and queries). Baselines include pseudo-relevance feedback (Lavrenko and Croft, 2001) and mutual information. They evaluate both CBOW and skip-gram `word2vec` embeddings (using default dimensionality and context window settings) but present only skip-gram results, noting "there was no big difference in retrieval performance between the two". The authors consider adding a fixed number of 1-10 expansion terms per query term and also compare two smoothing methods: linear Jelineck-Mercer vs. Dirichlet. Results show VEXP achieves higher Mean Average Precision (MAP) than other methods.

Roy et al. (2016) propose three approaches to utilize word embeddings in query expansion based on K-Nearest Neighbor (KNN). The first approach (i) computes the K nearest neighbors for each query term in the word embedding space. For each nearest neighbor, they calculate the average cosine similarity vs. all query terms, selecting the top-K terms according to average cosine score. The second approach (ii) reduces the vocabulary space considered by KNN by only considering terms appearing in the top M pseudo-relevant documents retrieved by the query. In the third approach (iii), an iterative (computationally expensive) pruning strategy is applied to reduce the number of nearest neighbors, assuming that nearest neighbors are similar to one another. Search is performed using unigram language model retrieval with Jelinek-Mercer smoothing. Baselines include no-expansion unigram and RM3 interpolated query expansion (Abdul-Jaleel et al., 2004) between unigram and RM1 relevance model (Lavrenko and Croft, 2001). Negative results show that the standard RM3 model performs better, suggesting that word embeddings do not yield improvements in this formulation.

Amer et al. (2016) investigate word embedding for personalized query expansion in the domain of social book search[15]. While personalized query expansion is not new (Chirita et al., 2007; Carmel et al., 2009), use of word embedding for personalization is novel. The proposed method consists of three steps: user modeling, term filtering and selection of expansion terms. A user is modeled as a collection of documents, and query terms are filtered to remove adjectives, which may lead to noisy expansion terms. For each remaining query term, similar to Roy et al. (2016)'s KNN approach, the top-K most similar terms are selected based on cosine similarity in the embedding space. Evaluation on the social book search task compares `word2vec` trained on personalized vs. non-personalized training sets. However, results show that expansion via word embeddings strictly hurts performance vs. no expansion at all, in contrast with findings of Roy et al. (2016) and Mitra and Craswell (2015). This may stem from training `word2vec` embeddings only on social book search documents. Results further suggest that

[15] `http://social-book-search.humanities.uva.nl`

personalized query expansion does not provide improvement over non-personalized query expansion using word embedding. The authors postulate that sparse training data for personalization is the main problem here and leave this for future work.

Rekabsaz et al. (2016) recommend choosing similar terms based on a global similarity threshold rather than by KNN because some terms should naturally have more similar terms than others. They choose the threshold by setting it such that for any term, the expected number of related terms within the threshold is equal to the average number of synonyms over all words in the language. This method avoids having to constrain or prune the KNN technique as in (Roy et al., 2016). They use multiple initializations of the `word2vec` skip-gram model to produce a probability distribution used to calculate the expected cosine similarity, making the measure more robust against noise. Experiments on TREC 6-8 and HARD 2005 incorporate this threshold-setting method into a translation language model (Berger and Lafferty, 1999) for ad-hoc retrieval and compare against both a language model baseline and a translation language model that uses KNN to select similar words. The threshold-based translation language model achieves the highest Mean Average Precision (MAP).

3.3 Sponsored Search

Grbovic et al. present a pair of studies in sponsored search (Grbovic et al., 2015a,b). In their first study, **Grbovic et al. (2015a)** propose `query2vec`, a two-layer architecture for search retargeting, where the upper layer models the temporal context of a query session using a `word2vec` skip-grammodel, and the lower layer models word sequences within a query using `word2vec` CBOW. They also introduce two incremental models: `ad-query2vec`, which incorporates the learning of ad click vectors in the upper layer by inserting them into query sequences after queries that occurred immediately prior to an ad click; and `directed ad-query2vec`, which uses past queries as context for a directed language model in the upper layer. The models are trained using 12 billion sessions collected on Yahoo search and evaluated offline using historical activity logs, where success is measured by the click-through rate of ads served. All three `query2vec` models show improvement over sponsored keyword lists and search retargeting using `word2vec` and query flow graph.

In their subsequent, longer study, **Grbovic et al. (2015b)** propose a method to train context and content-aware word embeddings. The first model they propose is `context2vec`. It treats a search session as a sentence and each query from the session as a word from the sentence. It uses `word2vec`'s skip-gram model. Queries with similar context will result in similar embeddings. The second model is `content2vec`. This method is similar to Le and Mikolov (2014)'s `ParagraphVector` in that it uses the query as a paragraph to predict the word in its content. The third model `context-content2vec`, similar to their earlier `query2vec`, combines `context2vec` and `content2vec` to build a two-layer model which jointly considers the query session context and the query context. To generate query rewrites, they use K-Nearest Neighbors (KNN) in the embedding space. They train embeddings on query content and search session data. They also incorporate ad click and search link click events as additional context to improve the query representation for a specific task. After finishing training embeddings, they evaluate them on an in-house data set and TREC Web Track dataset from 2009-2013. The best performance is achieved by `context-content2vec` with ad clicks and search link clicks in terms of NDCG and editorial grade. They also show that by considering ad clicks, the query rewrites generated can cover more bid terms.

3.4 Other Tasks

Community Question Answering (CQA). Learning of word embeddings coupled with category metadata for CQA is proposed by **Zhou et al. (2015)**. They adopt `word2vec`'s skip-gram model augmented with category metadata from online questions, with category information encoding the attributes of words in the question (see Zhang et al. (2016a) for another example of integrating categorical data with word embeddings). In this way, they group similar words based on their categories. They incorporate the category constraint into the original skip-gram objective function. After the word embedding is learned, they use Fisher kernel (FK) framework to convert the question into a fixed length vector (similar to Clinchant and Perronnin (2013) and Zhang et al. (2014)). To retrieve similar questions, they use the dot product of FVs to calculate the semantic similarities. For the experiment, they train the word embeddings on Yahoo! Answers and Baidu Zhidao for English and Chinese respectively. Results show that the category metadata powered model outperforms all the other baselines not using metadata. Future work might include exploring how to utilize other metadata information, such as user ratings, to train more powerful word embeddings.

Contextual Suggestion. For the task of context-aware venue recommendation, users can express a set of contextual aspects for their preferences, where each aspect has multiple contextual dimension terms, and each term has a list of related terms. The goal is to rank a list of venues by measuring how well each venue matches the user's contextual preferences. The traditional approach for this task, collaborative filtering, suffers from data sparsity. **Manotumruksa et al. (2016)** model user preferences using word embeddings, the only work we are aware of utilizing word embeddings for this task. They develop two approaches to model user-venue preferences and context-venue preferences using word embeddings. First, they infer a vector representation for each venue from user comments on it and model the user-venue preferences using the rated venues' vector representation in the user's profile. Second, they model the dimensions of each aspect in the context-venue preferences by identifying top similar terms for that dimension. To evaluate the user-venue preferences model, they first train word embeddings using `word2vec` skip-gram on Foursquare data giving venue comments. Then they calculate the cosine similarity between the vector representation of venue and the user and use it as a feature in the learning to rank system. They conduct experiments using Terrier on TREC 2013 and 2014 Contextual Suggestion tracks[16]. Results show that the system using word embeddings outperforms those without using embeddings. To evaluate the context-aware preference model, the authors use cosine similarity between the venue and each of the contextual aspect as a feature in the ranking system. They also incorporate venue-dependent features and user-venue preference features. Results on TREC 2015 Contextual Suggestion task show that the proposed new system outperforms the baseline which does not utilize user information and contextual preferences.

Cross-Lingual IR. A unified framework for monolingual and cross-lingual IR using word embeddings is proposed by **Vulic and Moens (2015)**. The monolingual model and results are presented in Section 3.2. Here, we discuss the authors' approach to learning bilingual embeddings and their findings for cross-lingual search. Their Bilingual Word Embeddings Skip-gram (BWESG) approach departs from prior work requiring sentence-aligned, parallel bilingual corpora (or bilingual dictionaries) to learn bilingual embeddings. Instead, only document-aligned *comparable* bilingual documents are needed. Their key idea is a *merge and shuffle* process in which matched documents from each language are first merged and sentence boundaries removed. Next, they randomly shuffle the words in the constructed pseudo-bilingual document. This assures that each word, regardless of language, obtains word *collocates* from both languages. While this approach is simple and able to benefit from a potentially vaster body of comparable vs. parallel corpora for training, random shuffling loses the precise local context windows exploited by `word2vec` training (which parallel corpora would provide), effectively setting context window size to the length of the entire document. The approach and experimental setup otherwise follows the monolingual version of the authors' method. Cross-lingual results for CLEF 2001-2003 Ad-hoc English-Dutch show that the embedding approach outperforms the unigram baseline and is comparable to the LDA baseline. As with monolingual results, the mixture models perform better, with a cross-lingual three way mixture of unigram, LDA, and embedding. No experiments are reported with parallel corpora for comparison, which would be interesting for future work.

Detecting Text Reuse. The goal of **Zhang et al. (2014)** is to efficiently retrieve passages that are semantically similar to a query, making use of hashing methods on word vectors that are learned in advance. Other than the given word vectors, no further deep learning is used. As in Clinchant and Perronnin (2013) and Zhou et al. (2015), they adopt the Fisher Kernel framework to convert variable-size concatenations of word embeddings to fixed length. However, this resulting fixed-length Fisher vector is very high-dimensional and dense, so they test various state-of-the-art hashing methods (e.g., Spectral Hashing (Weiss et al., 2009) and SimHash (Charikar, 2002)) for reducing the Fisher vector to a lower-

[16]`https://sites.google.com/site/treccontext/`

dimensional binary vector. Experiments are conducted on six collections including TIPSTER (Volumes 1-3), ClueWeb09-Cat-B, Tweets2011, SogouT 2.0, Baidu Zhidao, and Sina Weibo, with some sentences manually annotated for semantic similarity. Hashing methods that use Fisher vector representations based on word embeddings achieve higher precision-recall curves than hashing methods without vector representations and have comparable computational efficiency.

Domain-specific semantic similarity. Word embeddings have also shown promise in capturing semantic similarities in specific domains of information retrieval. **De Vine et al. (2014)** illustrate the ability of the `word2vec` skip-gram embedding to learn medical embeddings from patient records and medical journal abstracts. The embeddings produce semantic similarities that strongly correlate with medical expert evaluations. Specifically, they correlate better with expert-provided concept similarities than previous corpus-driven methods for semantic similarity such as latent semantic analysis (LSA). To achieve this, prior to training semantic embeddings, documents are preprocessed through a medical concept tagger, such that documents are transformed into sequences of medical concept identifiers. Thus, the authors also demonstrate the potential for word embeddings to be trained from structured ontologies rather than raw text. Performance differences between these approaches is left for future work. Additional future work includes evaluation of the effect of medical embeddings on the performance of medical IR systems.

Outlier Detection. The evaluation of **Le and Mikolov (2014)**'s proposed `ParagraphVector` (PV) model includes an unusual outlier detection IR task. Section 6.3 describes the PV model. In the task evaluation, each instance is a triplet of search result snippets, with two snippets deriving from the same query, and the third coming from a different query. The goal of the task is to detect the outlier snippet from each triplet. Results show that PV outperforms bag-of-words and bigram models.

Query classification. Using data from the KDD Cup 2005 Task[17], **Zamani and Croft (2016b)** propose an embedding method for categorizing queries. See Section 3.2 for description of Zamani and Croft (2016b)'s overall method and results for ad-hoc search. For query classification, given a training data pair $\langle query, category \rangle$, the authors first calculate the centroid vector of all query embedding vectors under a category. Then for a test query, they use the query embedding form and calculate the distance to the K nearest neighbor centroid vector. Finally, they use a softmax function over the set of calculated distances to determine the final set of categories for the test query. Because only embedding-based baselines are included in evaluation, it is unclear how the proposed approach would perform vs. traditional IR models.

Short text similarity. To capture the semantic similarity between a pair of short texts, **Kenter and de Rijke (2015)** propose a supervised machine learning algorithm using two different types of meta-features which utilize different publicly available word embeddings. Their first meta-feature resembles BM25, computing semantic text similarity based on word-level semantics (e.g., word embedding of each word). For their second meta-feature, they compute a matrix of cosine similarity in semantic (e.g., word embedding) space between each pair of terms of two sentences, quantizing similarities into three different bins. Along with these two features they introduce two minor text-level features. Each pairwise sentence represented by the features and labeled by the human annotator is used to train a Support Vector Machine (SVM) with Radial Basis Function (RBF) on the MSR Paraphrase Corpus dataset[18] (Dolan et al., 2004). Unlike many prior semantic similarity methods, this work requires no external resources (e.g., WordNet), and thus reduces the necessity of human feature engineering. The authors use both `word2vec` and GloVe (Pennington et al., 2014) embeddings as well as their own "auxiliary" trained embeddings on a dataset from INEX. Results suggest that their method using only publicly available embeddings, without any parameter tuning, outperforms state-of-the-art methods. Use of the auxiliary word embeddings in addition to public embeddings increases performance further. Future work could extend the approach to consider word order.

4 Background on Neural Networks

Assuming a readership familiar with IR but possibly less versed in neural network (NN) concepts, this section briefly introduces several such concepts which underlie surveyed work in Section 5. For further

[17]`http://www.kdd.org/kdd-cup/view/kdd-cup-2005`

[18]`https://www.microsoft.com/en-us/download/details.aspx?id=52398`

details on these concepts, please see the many informative resources on NN modeling cited in Section 1.

4.0.1 Neural Network (NN)

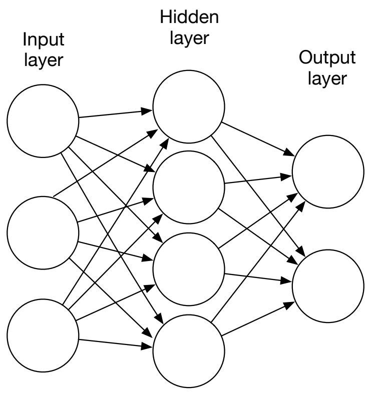

Figure 1: Feed-forward fully connected neural network

The neural models we will typically consider in this survey are *feed-forward* networks, which we refer to as neural networks (NNs) for simplicity. A simple example of such an NN is shown in **Figure 1**. Input features are extracted, or *learned*, by NNs using multiple, stacked fully connected layers. Each layer applies a linear transformation to the vector output of the last layer (performing an affine transformation). Thus each layer is associated with a matrix of parameters, to be estimated during learning. This is followed by element-wise application of a non-linear activation function. In the case of IR, the output of the entire network is often either a vector representation of the input or some predicted scores. During training, a loss function is constructed by contrasting the prediction with the ground truth available for the training data, where training adjusts network parameters to minimize loss. This is typically performed via the classic back-propagation algorithm (Rumelhart et al., 1988). For further details, see Goodfellow et al. (2016).

4.0.2 Auto-encoder

An *auto-encoder* neural network is an unsupervised model used to learn a representation for data, typically for the purpose of dimensionality reduction. Different from the typical NN, an auto-encoder is trained to reconstruct the input, and the output has the same dimension as the input. For more details, see Erhan et al. (2010) and Hinton and Salakhutdinov (2006).

4.0.3 Restricted Boltzmann Machine (RBM)

A Restricted Boltzmann Machine (RBM) is a stochastic neural network whose binary activations depend on its neighbors and have a probabilistic binary activation function. RBMs are useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning, topic modeling, etc. The RBM was originally proposed by Smolensky (1986) and further popularized by Nair and Hinton (2010).

4.0.4 Convolutional Neural Network (CNN)

In contrast to the densely-connected networks described above, a Convolutional Neural Network (CNN) (LeCun and Bengio, 1995) defines a set of linear filters (kernels) connecting only spatially local regions of the input, greatly reducing computation. These filters extract locally occurring patterns. CNNs are typically built following a "convolution+pooling" architecture, where a pooling layer following convolution further extracts the most important features while at the same time reducing dimensionality. We show a basic example of a CNN in **Figure 2**. CNNs were first established by their strong performance on image classification (Krizhevsky et al. (2012)), then later adapted to text-related tasks in NLP and IR (Collobert et al. (2011); Kalchbrenner et al. (2014); Kim (2014); Zhang and Wallace (2015)). As discussed in Section 6.3.1, Yang et al. (2016b) and Guo et al. (2016) question whether CNN models developed in computer vision and NLP to exploit spatial and positional information are equally-well suited to the IR domain.

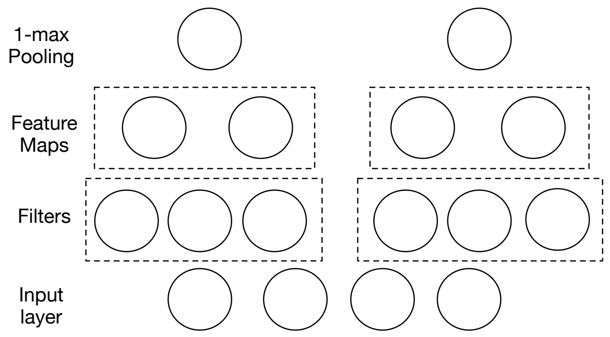

Figure 2: One-dimension Convolution Neural Network with only one convolution layer (two filters and two feature maps), followed by a 1-max-pooling layer.

4.0.5 Recurrent Neural Network (RNN)

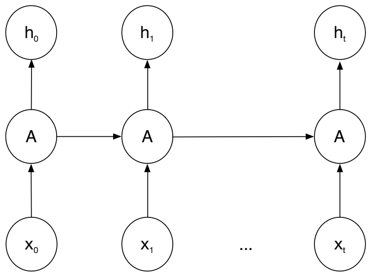

Figure 3: Recurrent Neural Network. x is the input, A is the computation unit shared across time steps, and h is the hidden state vector

A Recurrent Neural Network (RNN) (Elman (1990)) models sequential input, e.g., sequences of words in a document. We show a basic example of an RNN in **Figure 3**. Individual input units (e.g., words) are typically encoded in vector representations. RNNs usually read inputs sequentially; one can think of the input order as indexing "time". Thus the first word corresponds to an observation at time 0, the second at time 1, and so on. The key component of RNNs is a hidden state vector, which encodes salient information extracted from the input read thus far. At each step during traversal (at each time point t), this state vector is updated as a function of the current state vector and the input at time t. Thus each time point is associated with its own unique state vector, and when the end of a piece of text is reached, the state vector will capture the context induced by the entire sequence.

A technical problem with fitting the basic RNN architecture just described is the "vanishing gradient problem" (Pascanu et al., 2013) inherent to parameter estimation via back-propagation "through time". The trouble is that gradients must flow from later time steps of the sequence back to earlier bits of the input. This is difficult for long sequences, as the gradient tends to degrade, or "vanish" as it is passed backwards through time. Fortunately, there are two commonly used variants of RNN that aim to mitigate this problem (and have proven empirically successful in doing so): LSTM and GRU.

Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber (1997)) was the first approach introduced to address the vanishing gradient problem. In addition to the hidden state vector, LSTM introduces a *memory cell* structure, governed by three gates. An *input gate* is used to control how much the memory cell will be influenced by the new input; a *forget gate* dictates how much previous information in the memory cell will be forgotten; and an *output gate* controls how much the memory cell will influence the current hidden state. All three of these gates depend on the previous hidden state and the current input. For a more detailed description of LSTM and more LSTM variants, see Greff et al. (2015) and Graves (2013).

Bahdanau et al. (2014)'s **Gated Recurrent Unit (GRU)** is a more recent architecture, similar to the LSTM model but simpler (and thus with fewer parameters). Empirically, GRUs have been found to perform comparably to LSTMs, despite their comparative simplicity (Chung et al., 2014). Instead of the memory cell used by LSTMs, an *update gate* is used to govern the extent to which the hidden gate will be updated, and a *reset gate* is used to control the extent to which the previous hidden state will influence the current state.

4.0.6 Attention

The notion of *attention* has lately received a fair amount of interest from NN researchers. The idea is to imbue the model with the ability to *learn* which bits of a sequence are most important for a given task (in contrast, e.g., to relying only on the final hidden state vector).

Attention was first proposed in the context of neural machine translation model by Bahdanau et al. (2014). The original RNN model (Sutskever et al. (2014)) for machine translation encodes the source sentence into a fixed-length vector (by passing an RNN over the input, as described above). This is then accepted as input by a *decoder* network, which uses the single encoded vector as the only information pertaining to the source sentence. This means all relevant information required for the translation must be stored in a single vector – a difficult aim.

The attention mechanism was proposed to alleviate this requirement. At each time step (as the decoder generates each word), the model identifies a set of positions in the source/input sentence that is most relevant to its current position in the output (a function of its index and what it has generated thus far). These positions will be associated with corresponding state vectors. The current contextualizing vector (to be used to generate output) can then be taken as a sum of these, weighted by their estimated relevance. This attention mechanism has also been used in image caption generation (Xu et al. (2015)). A similar line of work includes *Neural Turing Machines* by Graves et al. (2014) and *Memory Networks* by Weston et al. (2014).

5 Neural Network Approaches to IR

Availability of code and existing embeddings from `word2vec` (Mikolov and Dean, 2013) and GloVe (Pennington et al., 2014) (Tables 4 and 5, respectively) have provided a key avenue for work on Neural IR, especially for extending traditional IR models to integrate word embeddings. Section 3 surveyed studies in this space. In contrast, we now survey conceptually different approaches which directly incorporate word embeddings within NN models, reflecting a more significant shift toward pursuing *end-to-end* NN architectures in IR. We organize surveyed studies by IR task (see **Table 2**). For additional background on NN concepts underlying the approaches surveyed in this section, see Section 4.

5.1 Ad-hoc Retrieval

As discussed by Guo et al. (2016), neural models for this tend to follow one of two general architectures, which we depict in **Figure 4**. In the first (Figure 4A), queries and documents are separately passed through neural networks (with shared parameters) and scores are generated based on the combination (e.g., concatenation) of the respective outputs. By contrast, Figure 4B depicts an approach in which a joint representation for queries and documents is constructed and this is run through a neural network. Guo et al. (2016)'s work is further discussed below.

Huang et al. (2013) propose a Deep Structured Semantic Model (DSSM) for ad-hoc web search. DSSM was one of the pioneer models incorporating click-through data in deep NNs and has been built on by a variety of others (Shen et al., 2014b,a; Ye et al., 2015; Mitra, 2015; Mitra and Craswell, 2015). Notably, documents are indexed only by title text rather than the entire body text. The query and the document are first modeled as two high dimensional term vectors (i.e., a bag-of-words representation). DSSM learns a representation of documents and queries via a feed-forward NN to obtain a low-dimensional vector projected within a latent semantic space. The document ranking is then trained within the DSSM architecture by maximizing the conditional likelihood of query given document. More specifically, the authors estimate this conditional likelihood by a softmax function applied on the cosine similarity between the corresponding semantic vector of documents and queries. Given large vocabularies and the need to support large-scale training, the authors propose a word hashing method which transforms the high-dimensional term vector of the query/document to a low-dimensional letter-trigram vector. This

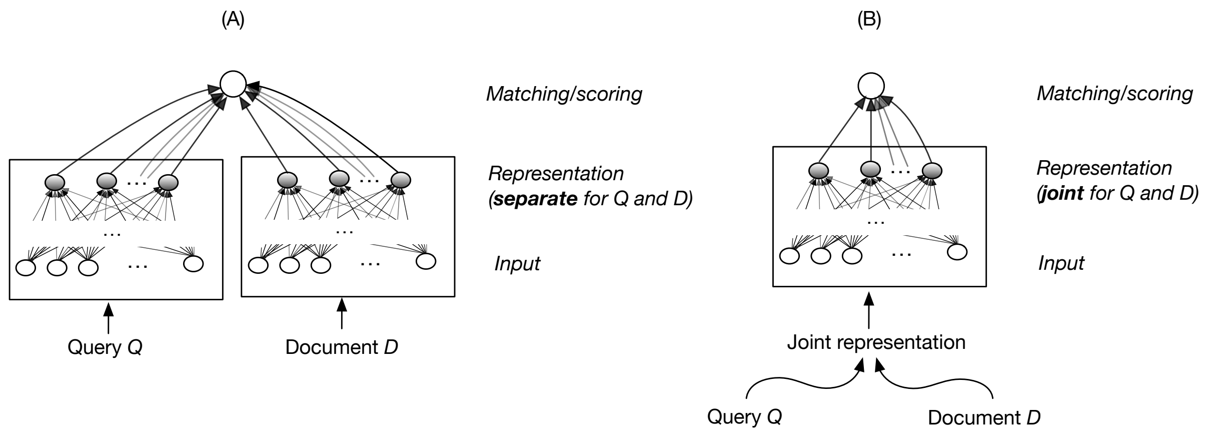

Figure 4: Two basic neural architectures for scoring the relevance of queries to documents. On the left (A), the query and document are independently run through mirror neural models (typically this would be a Siamese architecture (Bromley et al., 1993), in which weights are shared between the networks); relevance scoring is then performed by a model operating over the two induced representations. On the right (B), one first constructs a joint representation of the query and document pair and then runs this joint input through a network. The architectures differ in their inputs and the semantics of the hidden/representational layers. This figure is inspired by Figure 1 in Guo et al. (Guo et al., 2016).

lower dimensional vector is then considered as input to the feed-forward NN. Trigram hashing also helps to address out of vocabulary (OOV) query terms not seen in training data. The authors do not discuss how to mitigate hashing collisions; while they show that such collisions are relatively rare (e.g., 0.0044% for 500K vocabulary size), this stems in part from indexing document titles only and not body text. Later work by Guo et al. (2016) performs two further experiments with DSSM: indexing only document titles vs. indexing entire documents (i.e., full-text search). Guo et al. (2016)'s results indicate that full-text search with DSSM does not perform as well as traditional IR models.

Shen et al. (2014b) extend DSSM (Huang et al., 2013) by introducing a CNN with max-pooling in the DSSM architecture (C-DSSM). It first uses word hashing to transform each word into a vector. A convolutional layer then projects each word vector within a context window to a local contextual feature vector. It also incorporates a max-pooling layer to extract the most salient local features to form a fixed-length global feature vector for queries and web documents. The main motivation for the max-pooling layer is that because the overall semantic meaning of a sentence is often determined by a few key words, simply mixing all words together (e.g., by summing over all local feature vectors) may introduce unnecessary divergence and hurt the overall semantic representation effectiveness. This is a key difference between DSSM and C-DSSM.

Both DSSM (Huang et al., 2013) and C-DSSM (Shen et al., 2014b) fail to capture the contextual information of the queries and the documents. To address this, **Shen et al. (2014a)** propose a Convolutional Latent Semantic Model (CLSM) built atop DSSM. CLSM captures the contextual information by a series of projections from one layer to another in a CNN architecture (LeCun and Bengio, 1995). The first layer consists of a word n-gram followed by a letter trigram layer where each word n-gram is composed of its trigram representation, a form of the word hashing technique developed in DSSM. Then, a convolution layer transforms these trigrams into contextual feature vectors using the convolution matrix W_c, which is shared among all the word n-grams. Max-pooling is then performed against each dimension on a set of contextual feature vectors. This generates the global sentence level feature vector v. Finally, using the non-linear transformation $tanh$ and the semantic projection matrix W_s, they compute the final latent semantic vector for the query/document. The parameters W_c and W_s are optimized to maximize the same loss function used by Huang et al. (2013) for DSSM. Even though CLSM introduces the word n-gram to capture the contextual information, it suffers the same problems as DSSM including scalability. For example, CLSM performs worse when trained on a whole document than when trained on only the document title, as shown in experiments reported by Guo et al. (2016).

Task	**Studies**
Ad-hoc Retrieval	BP-ANN (Yang et al. (2016a)), CDNN (Severyn and Moschitti (2015)), C-DSSM (Shen et al. (2014b)), CLSM ((Shen et al., 2014a)), DSSM (Huang et al. (2013)), DRMM (Guo et al. (2016)), GDSSM (Ye et al. (2015)), Gupta et al. (2014), Li et al. (2014), Nguyen et al. (2016), QEM (Sordoni et al. (2014))
Conversational Agents	DL2R (Yan et al. (2016))
Proactive Search	Luukkonen et al. (2016)
Query Auto-completion	Mitra (2015); Mitra and Craswell (2015)
Query Suggestion	Sordoni et al. (2015)
Question Answering	BLSTM (Wang and Nyberg (2015)), CDNN (Severyn and Moschitti (2015)), DFFN (Suggu et al. (2016)), DL2R (Yan et al. (2016)), Yu et al. (2014)
Recommendation	Gao et al. (2014), Song et al. (2016)
Related Document Search	Salakhutdinov and Hinton (2009)
Result Diversification	Xu and Cheng (2016)
Sponsored Search	Zhang et al. (2016a)
Summarizing Retrieved Documents	Lioma et al. (2016)
Temporal IR	Kanhabua et al. (2016)

Table 2: IR tasks solved by Neural Network methods.

Nguyen et al. (2016) propose two high level views of how to incorporate a knowledge base (KB) graph into a ranking model like DSSM (Huang et al., 2013). To the best of our knowledge, this is one of the first IR studies which tries to incorporate KB into deep neural structure. The authors' first model exploits knowledge bases to enhance the representation of a document-query pair and its similarity score by exploiting concept embeddings learned from the KB distributed representation (Xu et al., 2014; Faruqui et al., 2014) as input to deep NNs like DSSM. The authors argue that this hybrid representation of the distributional semantic (namely, word embeddings) and the symbolic semantics (namely, concept embeddings taking into account the graph structure) would enhance document-query matching. Their second model uses the knowledge resource is as an intermediate component that helps to translate the deep representation of the query towards the deep representation of the document with respect to the ad-hoc IR task. Strong empirical evidence is still needed to demonstrate that adding KB indeed benefits the deep neural architecture for capturing semantic similarity.

Although DSSM (Huang et al., 2013), C-DSSM (Shen et al., 2014b) and CLSM (Shen et al., 2014a) achieve superior performance over traditional latent semantic models (e.g., PLSA, LDA), these models make some strong assumptions about clicked $\langle query, document \rangle$ pairs. According to **Ye et al. (2015)** these three assumptions are: i) each clicked $\langle query, document \rangle$ pair is equally weighted; ii) each clicked $\langle query, document \rangle$ pair is a 100% positive example; and iii) each click is solely due to the semantic similarity. The authors relax these assumption and propose two generalized extensions to DSSM: GDSSM1 and GDSSM2. While DSSM models the probability of a document being clicked given a query and the semantic similarity between document and query, GDSSM1 uses more information in its loss function, incorporating the number of users seeing and proportion clicking for each query-document pair. GDSSM2 conditions on lexical similarity in addition to semantic similarity. Potential future work includes developing a similar loss function as used in CLSM (Shen et al., 2014b).

Li et al. (2014) propose a document re-ranking model based on fine-grained features derived from user contextual data, such as the history of user queries and clicked documents. The major contribution of their work is to model user dissatisfaction from the click-through data and develop semantic similarity between previous and current queries based upon this dissatisfaction. The authors model contextual information as a combination of "global" (i.e., query-level) and "local" (single URL-level) components. Their evaluation involves a re-ranking task in which the rank of an original search engine is used as an input feature alongside the contextual features derived by their neural models. They evaluate XCode (internally developed by the authors), DSSM (Huang et al., 2013), and C-DSSM (Shen et al., 2014b) as models for deriving these contextual features and find that DSSM offers the highest performance, followed by C-DSSM. Though they expected C-DSSM to offer the highest performance, they note that C-DSSM could only be trained on a small dataset with bounded size, in contrast to DSSM, which could be trained on a larger dataset. Additionally, they note that the performance difference could be due to imperfect tuning of model parameters, such as sliding window size for C-DSSM. Nevertheless, contextual features derived using both DSSM and C-DSSM offer performance benefits for re-ranking. Potential future work includes incorporating user profiles in re-ranking.

Guo et al. (2016)'s Deep Relevance Matching Model (DRMM) is one of the first NN IR models to show improvement over traditional IR models (though comparison is against bag-of-words approaches rather than any term proximity baselines, such as Metzler and Croft (2005)'s Markov Random Field model). The authors hypothesize that deep learning methods developed and/or commonly applied in NLP for semantic matching may not be well-matched with ad-hoc search, which is most concerned with relevance matching. They articulate three key differences they perceive between semantic and relevance matching:

1. Semantic matching looks for semantic similarity between terms; relevance matching puts more emphasis on exact matching.

2. Semantic matching is often concerned with how composition and grammar help to determine meaning; varying importance among query terms is more crucial than grammar in relevance matching.

3. Semantic matching compares two whole texts in their entirety; relevance matching might only compare parts of a document to a query.

The raw inputs to the DRMM are term embeddings. The first layer consists of *matching histograms* of each query term's cosine similarity scores with each of the document terms. On top of this histogram layer is an NN outputting a single node for each query term. These outputs are multiplied by query term importance weights calculated by a *term gating network* and then summed to produce the final predicted matching score for the query/document pair. The whole network is trained using a margin ranking loss function. Ad-hoc search experiments are reported on TREC Robust04 and ClueWeb09-Cat-B. Baselines include: i) traditional retrieval models such as BM25 and query likelihood; ii) representation-focused NN models, including DSSM (Huang et al., 2013) and C-DSSM (Shen et al., 2014b) (indexing titles vs. full documents), ARC-I (Hu et al., 2014); and iii) interaction-focused NN models, such as ARC-II (Hu et al., 2014) and MatchPyramid (Pang et al., 2016b). OOV terms are represented by random vectors (as in (Kenter and de Rijke, 2015)), effectively allowing only exact matching. Results show that as CBOW dimension increases (50, 100, 300 and 500), DRRM first increases then slightly drops. DRRM using IDF and Log-Count Histogram (LCH) also significantly outperforms all baselines. In addition, all representation-focused and interaction-focused baselines are not competitive with traditional retrieval models, supporting the authors' hypothesis that deep matching models focused on semantic matching may not be well-suited to ad-hoc search. For future work, they propose using click-through log data to augment training.

Yang et al. (2016a) propose a document ranking model which decides if using term proximity ranking for a query would be beneficial based on its features. Term proximity (TP) requires especially large indexes and is computationally expensive. The proposed model seeks to return improved rankings while also reducing the overhead incurred by using TP only when it can be helpful. Yang et al. (2016a) use the popular TP ranking functions BM25TP (Rasolofo and Savoy, 2003), Markov Random Field (MRF) (Metzler and Croft, 2005), and "EXP" (Tao and Zhai, 2007). If a query achieves better MAP results using BM25TP or EXP over BM25 or using MRF over TF (Metzler and Croft, 2005), its features are labeled 1, otherwise 0. The results are used to train a Back-Propagation Artificial Neural Network (BP-ANN) (Rumelhart et al., 1988), which is used to build the proposed selective model. A combination of statistical methods (ranksum, z-score, χ -square), a decision tree, and a feature weight algorithm (relief) are employed to find important features for BM25TP, MRF, and EXP to construct the BP-ANN. The experimental evidence shows that consistently using TP achieves significantly better rankings. However, the results suggest that selectively using TP returns slightly better rankings and has better throughput than when always using TP. Future work could include introducing more important features capturing term proximity, as well as training a collection of models, each designed for a particular query type, for query-dependent feature generation.

5.1.1 Query Expansion

Sordoni et al. (2014) propose a supervised Quantum Entropy Minimization (QEM) approach for finding semantic representation of *concepts*, such as words or phrases, for query expansion. The authors suggest that text sequences should not lie in the same semantic space as single terms because their information content is higher. To this end, concepts are embedded in rank-one matrices while queries and documents are embedded as a mixture of rank-one matrices. This allows documents and queries to lie in a larger space and carry more semantic information than concepts, thereby achieving greater semantic resolution. As such, QEM strives for an increased representational power vs. existing approaches relying only on vector representations (for example, combining individual word embeddings by simple arithmetic operations like averaging to learn semantic representations for larger textual units; see broader discussion in Section 6.3). For learning parameters of *density matrices*, QEM's gradient updates are a refinement of updates in both Bai et al. (2009)'s Supervised Semantic Indexing (SSI) and Rocchio (Rocchio, 1971); a query concept embedding moves toward its nearest relevant documents concepts and away from its nearest non-relevant documents concepts. This has the effect of selecting which document concepts the query concept should be aligned with and also leads to a refinement of Rocchio: the update direction for query expansion is obtained by weighting relevant and non-relevant documents according to their similarity to the query. Due to lack of public query logs, QEM is trained on Wikipedia *anchor logs* (Dang and Croft, 2010). Experiments conducted on ClueWeb09-Cat-B with TREC 2010-2012 Web Track topics show QEM outperforms Gao et al. (2010)'s concept translation model (CTM) (statistically significant differences), and SSI (occasionally statistically significant differences). QEM notably achieves improved precision at top-ranks. Also notable is QEM's ability to find useful expansion terms for longer queries due to higher semantic resolution. Additional preliminary experiments with Weston et al. (2010)'s *Weighted Approximate-Rank Pairwise* loss yielded further improvements for QEM over baselines. Proposed future work includes exploring ways to automatically set appropriate embedding dimensions. In addition, the difficult maximization when computing the embeddings for queries and documents relies on a linear Taylor expansion for approximation, which might be further improved. Better approximations may exist than the linear approximation used here, particularly for the scoring function.

Gupta et al. (2014) explore query expansion in mixed-script IR (MSIR), a task in which documents and queries in non-Roman written languages can contain both native and transliterated scripts together. Stemming from the observation that transliterations generally use Roman letters in such a way as to preserve the original-language pronunciation, the authors develop a method to convert both scripts into a bilingual embedding space. Therefore, they convert terms into feature vectors of the count of each Roman letter. An *auto-encoder* is then trained for query expansion by feeding in the concatenated native term feature vector and transliterated term feature vector. Experiments are conducted on shared task data from FIRE (Section 3.2). Results suggest that their method significantly outperforms other state-of-the-art methods. For future work, while the query projection into abstract space is performed here using matrix multiplication online, this could be further optimized. An ablation study of features would help elucidate the relative importance of different features (abstract feature space) and to determine which contribute most to model performance. Source code for the model is shared online (see Table 5).

5.2 Query Auto-completion

Mitra (2015) learns embedding vectors for query reformulation based on query logs, representing query reformulation as a vector using Shen et al. (2014b)'s Convolution Latent Semantic Model (CLSM). Two sets of features are developed. The first feature set captures topical similarity via cosine similarity between the candidate embedding vector and embedding vectors of some past number of queries in the same session. The second set of *reformulation features* captures the difference between the candidate embedding vector and that of the immediately preceding query. Other features used include non-contextual features, such as most popular completion, as well as contextual features, such as n-gram similarity features and pairwise frequency features. LambdaMART (Wu et al., 2010) is trained on click-through data and features. Results suggest that embedding features give considerable improvement over Most Probable Completion (MPC) (Bar-Yossef and Kraus, 2011), in addition to other models lacking the embedding-based features.

Whereas traditional query auto-completion (QAC) methods perform well for head queries having abundant training data, prior methods often fail to recommend completions for tail queries having less usual prefixes, including both probabilistic algorithms (Bar-Yossef and Kraus, 2011; Cai et al., 2014; Shokouhi and Radinsky, 2012) and learning-based QAC approaches (Cai and de Rijke, 2016; Jiang et al., 2014). To address this, **Mitra and Craswell (2015)** develop a query auto-completion procedure for such rare prefixes which enables QAC even for query prefixes never seen in training. The authors mine the most popular query suffixes (e.g., n-grams which appear at the end of a query) and append them to the user query prefixes, thus generating possible synthetic candidate solutions. To recommend possible query expansion, they use LambdaMART (Wu et al., 2010) with two sets of ranking features. The first feature set is the frequency of query n-grams in the search log. The second feature set is generated by training CLSM on the prefix-suffix dataset, sampling queries in the search logs and segmenting each query at every possible word boundary. Results on the AOL search log show significant improvement over MPC (Bar-Yossef and Kraus, 2011). The authors also find n-gram features to be more important than CLSM features in contributing to model performance.

5.3 Question Answering: Answer Sentence Selection

Methods proposed in this section all evaluate on Wang et al. (2007)'s dataset derived from TREC QA 8-13.

Wang and Nyberg (2015) use a stacked bi-directional LSTM (BLSTM) to read a question and answer sequentially and then combine the hidden memory vectors from LSTMs of both question and answer. They use mean, sum and max-pooling as features. This model needs to incorporate key-word matching as a feature to outperform previous approaches that do not utilize deep learning. They use BM25 as the key-word matching feature and use Gradient boosted regression tree (GBDT) Friedman (2001) to combine it with the LSTM model.

Severyn and Moschitti (2015) propose a Convolutional Deep Neural Network (CDNN) to rank pairs of short texts. Their deep learning architecture has 2 stages. The first stage is a sentence embedding model using a CNN to embed question and answer sentences into intermediate representative vectors, which are used to compute their similarity. The second stage is a NN ranking model whose features include intermediate representative sentence vectors, similarity score, and some additional features such as word overlap between sentences. Results show improvement of about 3.5% in MAP vs. results reported by Yu et al. (2014), in which a CNN is used followed by logistic regression (LR) to rank QA pairs. The authors attribute this improvement to the larger width (of 5) of the convolutional filter in their CNN for capturing long term dependencies, vs. the unigram and bigram models used by Yu et al. (2014). Beyond the similarity score, their second stage NN also takes intermediate question and answer representations as features to constitute a much richer representation. Yu et al. (2014) do not include such intermediate question and answer representations as features to their LR approach.

Yang et al. (2016b)'s approach starts by considering the interaction between question and answer at the word embedding level. They first build a question-answer interaction matrix using pre-trained embeddings. They then use a novel value-shared weight CNN layer (instead of a position-shared CNN) in order to induce a hidden layer. The motivation for this is that different matching ranges between a question term and answer will influence the later ranking score differently. After this, they incorporate an *attention network* for each question term to explicitly encode the importance of each question term and produce the final ranking score. They rank the answer sentences based on the predicted score and calculate MAP and MRR. Whereas Severyn and Moschitti (2015) and Wang and Nyberg (2015) need to incorporate additional features in order to achieve comparative performance, Yang et al. (2016b) do not require any feature engineering.

5.4 Community Question Answering (CQA)

Suggu et al. (2016) propose a Deep Feature Fusion Network (DFFN) to exploit both hand-crafted features (HCF) and deep learning based systems for Answer Question Prediction. Specifically, query/answer sentence representations are embedded using a CNN. A single feature vector of 601 dimensions serves as input to a second stage fully-connected NN. Features include sentence representations, HCF (e.g., TagMe, Google Cross-Lingual Dictionary (GCD), and Named Entities (NEs)), similarity measures between questions and answers, and metadata such as an answer author's reputation score. The output of the second stage NN is a score predicting the answer quality. They compare their approach to the top two best performing HCF based systems from SemEval 2015 and a pure deep learning system. For SemEval 2016, DFFN was compared with their corresponding top two best performing system. Results show that DFFN performs better than the top systems across all metrics (MAP, F1 and accuracy) in both SemEval 2015 and 2016 datasets. The authors attribute this to fusing the features learned from deep learning and HCF, since some important features are hard to learn automatically. As an example, question and answer often consists of several NEs along with variants, which are hard to capture using deep learning. However, NEs can be extracted from QA and their similarity used as a feature. Their use of HCF was also motivated by the many available similarity resources, such as Wikipedia, GCD, and click-through data, which could be leveraged to capture richer syntactic and semantic similarities between QA pairs.

5.5 Recommendation

Due to difficulty modeling temporal behavior in recommendation systems, many existing techniques treat users' interests as static long-term features. **Song et al. (2016)** are mainly concerned with how to incorporate short-term users' interest to improve the recommendation quality of news, where freshness is particularly important. Based on DSSM (Huang et al., 2013), they propose a temporal deep semantic structured model (TDSSM). It has two views represented by two neural networks. The item view contains the "implicit feedback of items" and the user view contains "user's query history." An RNN is used to model short-term temporal user interests. They also expand their model to a multi-rate-TDSSM (MR-TDSSM) model, which can take different granularities of time as input: slow-rate RNNs take daily (user, news) clicks, whereas fast-rate RNNs take weekly clicks. Data used consists of (user, news) clicks from a commercial News recommendation system of user news click history over six months. They compare the performance of TDSSM with several traditional recommendation algorithms and find that TDSSM and MR-TDSSM outperform previous methods not using deep learning. Regarding future work, the authors propose to apply the proposed model to different recommendation tasks and explore an attention-based memory network model.

Gao et al. (2014) propose using DSSM for both automatic highlighting of relevant keywords in documents and recommendation of alternative relevant documents based upon these keywords. They evaluate their framework based on what they call *interestingness* tasks, derived from Wikipedia anchor text and web traffic logs. They find that feeding DSSM derived features into a supervised classifier for recommendation offers state-of-the-art performance and is more effective than simply computing distance in the DSSM latent space. Future work could incorporate complete user sessions, since prior browsing and interaction history recorded in the session provide useful additional signals for predicting interestingness. This signal might be modeled most easily by using an RNN.

5.6 Other Tasks

Conversational Agents. An automatic conversation response system called Deep Learning-to-Respond (DL2R) is proposed by **Yan et al. (2016)**. They train and test on 10 million $\langle posting, reply \rangle$ pairs of human conversation web data from various sources, including microblog websites, forums, Community Question Answering (CQA) bases, etc. For a given query they reformulate it using other contextual information and retrieve the most likely candidate reply. They model the total score as a function of three scores: query-reply, query-posting, and query-context, each fed into a neural network consisting of bi-directional LSTM RNN layers, convolution and pooling layers, and several feed-forward layers. The strength of DL2R comes from the incorporation of reply, posting, and context with the query. However, since DL2R is a supervised learning algorithm and training data is very scarce in practice, a possible future direction would be to develop an unsupervised method for calculating similarity of sentence-pairs.

Proactive Search. In the only work we know of investigating Neural IR for proactive retrieval, **Luukkonen et al. (2016)** propose LSTM-based text prediction for query expansion. Intended to better support proactive intelligent agents such as Apple's *Siri*, *Ok Google*, etc., the LSTM is used to generate sequences of words based on all previous words written by users. A beam search is used to prune out low probability sequences. Finally, words remaining in the pruned tree are used for query expansion. Because LSTMs have the ability to represent a given text as well as to predict text, the proposed model outperforms traditional models (Glowacka et al., 2013). The authors provide several possible future directions, such as using the model to automatically suggest different continuations for user text as it is written, as done in the Reactive Keyboard (Darragh et al., 1990) and akin to query auto-completion in search engines. User studies are also needed to test the system's effectiveness in the context of users' real-world tasks.

Query Suggestion. We know of no work using RNNs for query suggestion prior to **Sordoni et al. (2015)**'s training a hierarchical GRU model to generate context-aware suggestions. They first use a GRU layer to encode the queries in each session into vector representations, then build another GRU layer on sequences of query vectors in a session and encode the session into a vector representation. The model learns its parameters by maximizing the log-likelihood of observed query sessions. To generate query suggestions, the model uses a GRU decoder on each word conditioned on both the previous words generated and the previous queries in the same session. The model estimates the likelihood of a query suggestion given the previous query. A learning-to-rank system is trained to rank query suggestions, incorporating the likelihood score of each suggestion as a feature. Results on the AOL query log show that the proposed approach outperforms several baselines which use only hand-crafted features. The model is also seen to be robust when the previous query is noisy. Moreover, because the model can generate synthetic queries, it can effectively handle long tail queries. However, only previous queries from the same session are used to provide the contextual query suggestion; the authors do not utilize click-through data from previous sessions. Because click-through data provides important feedback for synthetic query suggestion, incorporating such click-through data from previous sessions represents a possible direction for future work.

Related Document Search. Semantic hashing is proposed by **Salakhutdinov and Hinton (2009)** to map semantically similar documents nearby to one another in hashing space, facilitating easy search for similar documents. Multiple layers of Restricted Boltzmann Machines (RBMs) are used to learn the semantic structure of documents. The final layer is used as a hash code that compactly represents the document. The lowest layer is simply word-count data and is modeled by the Poisson distribution. The hidden layers are binary vectors of lower dimensions. The deep generative model is learned by first pre-training the RBMs one layer at a time (from bottom to top). The network is then "unrolled", i.e., the layers are turned upside down and stacked on top of the current network. The final result is an *auto-encoder* that learns a low-dimensional hash code from the word-count vector and uses that hash code to reconstruct the original word-count vector. The auto-encoder is then *fine-tuned* by back-propagation. Results show that semantic hashing is much faster than locality sensitive hashing (Andoni and Indyk, 2006; Datar et al., 2004) and can find semantically similar documents in time independent of document collection size. However, it imposes difficult optimization and a slow training mechanism, reducing applicability to large-scale tasks (Liu et al., 2012) and suggesting possible future work. In addition, because relevance judgments were not available, document corpora with category labels were used instead, assuming relevance if a query and document had matching category labels.

Result Diversification. Prior state-of-the-art methods for diversifying search results include the Relational Learning-to-Rank framework (R-LTR) (Zhu et al., 2014) and the Perceptron Algorithm using Measures as Margins (PAMM) (Xia et al., 2015). These prior methods either use a heuristic ranking model based on a predefined document similarity function, or they automatically learn a ranking model from predefined novelty features often based on cosine similarity. In contrast, **Xu and Cheng (2016)** takes automation a step further, using Neural Tensor Networks (NTNs) to learn the novelty features themselves. The NTN architecture was first proposed to model the relationship between entities in a knowledge graph via a bilinear tensor product (Socher et al., 2013). The model here takes a document and a set of other documents as input. The architecture uses a tensor layer, a max-pooling layer, and a linear layer to output a document novelty score. The NTN augmentations of R-LTR and PAMM perform at least as well as those baselines, showing how the NTN can remove the need for manual design of functions and features. It is not clear whether using a full tensor network works much better than just using a single slice of the tensor.

Sponsored Search. Predicting click-through rate (CTR) and conversion rate from categorical inputs such as region, ad slot size, user agent, etc., is important in sponsored search. **Zhang et al. (2016a)** propose the first neural approach we know of to predict CTR for advertising. An interesting aspect of the work is learning vector embeddings of categorical variables in a similar way to natural language terms (see Zhou et al. (2015) for another example of integrating categorical data with word embeddings). The authors develop two deep learning approaches to this problem, a Factorization Machine supported Neural Network (FNN) and Sampling-based Neural Network (SNN). The Factorization Machine is a non-linear model that can efficiently estimate feature interactions of any order even in problems with high sparsity by approximating higher order interaction parameters with a low-rank factorized parameterization. The use of FM-based bottom layer in the deep network, therefore, naturally solves the problem of high computational complexity of training neural networks with high-dimensional binary inputs. The SNN is augmented either by a sampling-based Restricted Boltzmann Machine (SNN-RBM) or a sampling-based Denoising Auto-Encoder (SNN-DAE). The main challenge is that given many possible values of several categorical fields, converting them into dummy variables results in a very high-dimensional and sparse input space. For example, thirteen categorical fields can become over 900,000 binary inputs in this problem. The FNN and SNN reduce the complexity of using a neural network on such a large input by limiting the connectivity in the first layer and by pre-training by selective sampling, respectively. After pre-training, the weights are *fine-tuned* in a supervised manner using back-propagation. Evaluation focuses on the tuning of SNN-RBM and SNN-DAE models and their comparison against logistic regression, FM and FNN, on the *iPinYou* dataset[19] (Liao et al., 2014) with real user click data, divided into five test sets. Results on each test set show one of the proposed methods performs best, though the baselines are often close behind and twice take second place. The authors also find a diamond-shape architecture is better than increasing, decreasing, or constant hidden-layer sizes and that dropout works better than L2 regularization. For future work, the model performance might be improved by a momentum method for handling the curvature problems in the DNN training objectives without using complex second-order methods. In addition, the partial connection in the bottom layer could be extended to higher layers as partial connectivities have many advantages, such as lower complexity and higher generalization ability. Though this method can extract non-linear features, it is only very effective when dealing with advertisements without images. Consequently, further research on multi-modal sponsored search to model images and text would be useful to pursue.

Summarizing Retrieved Documents. While IR is typically understood to stop at retrieving relevant documents, **Lioma et al. (2016)** instead propose to synthesize retrieved documents, generating a new document for the user. They use an LSTM, trained using Torch[20], to generate the synthetic documents.

[19] `http://data.computational-advertising.org`
[20] `https://github.com/jcjohnson/torch-rnn`

The LSTM takes the concatenated text of the query and its known relevant documents using word embeddings as input. However, rather than taking the whole text of a document, the authors extract a context window of $\pm n$ terms around every query term. To evaluate synthetic documents, the authors collect user evaluations through CloudFlower[21], defining four queries per job. Each user is presented the query text and four wordclouds, generated from the synthetic document and three random relevant documents. Though deeper semantics are lost, wordclouds are selected for ease of the users, and each user is tasked with ordering the wordclouds from most to least relevant to the query. Experimental results suggest that the deep learned synthetic documents were on average considered to be more relevant than existing indexed relevant documents. For future work, Lioma et al. (2016) plan to experiment with word-level RNNs to produce better quality synthetic documents, as character-level RNNs have been criticized for producing noisy pseudo-terms.

Temporal IR. Temporal IR is concerned with the temporal relevance of documents. For example, some queries are related to real events in the world and are expected to be matched to documents also pertaining to the same event more so than to static elements of the query. **Kanhabua et al. (2016)** use a deep NN to classify when a document is related to a temporal event and to identify the type of temporal event (anticipated, breaking, commemorative, meme, or ongoing). Hand-coded temporal query log features are used as input, such as burst length, average frequency, existence of person or location entities, auto-correlation, and click entropy. Despite manually-labeled target data being quite sparse, their deep network outperforms shallower baselines like Naive Bayes (Rish, 2001), Multi-layer Perceptron (MLP), and LibSVM (Chang and Lin, 2011). However, it is unclear how hand-coded features would be obtained for dynamic events in order to detect them.

6 Discussion

6.1 Word Embedding

Use of word embeddings involves many design decisions. Should one use `word2vec` (CBOW or skip-gram) (Mikolov and Dean, 2013; Mikolov et al., 2013), GloVe (Pennington et al., 2014), or something else (such as count-based embeddings)? Can embeddings from multiple sets be selected dynamically or combined together (Neelakantan et al., 2014; Zhang et al., 2016b)? How should hyper-parameters be set (e.g., dimensionality of embedding space, window size, etc.)? What training data/corpora should be used? Presumably larger training data is better, along with in-domain data similar to the test data on which the given system is to be applied, but how does one trade-off greater size of out-of-domain data vs. smaller in-domain data? How might they be best used in combination? Does performance vary much if we simply use off-the-shelf embeddings (e.g., from `word2vec` or GLoVe) vs. re-training embeddings for a target domain, either by *fine-tuning* (Mesnil et al., 2013) off-the-shelf embeddings or re-training from scratch? How much does task (e.g., ad-hoc search vs. document classification) or downstream architecture matter, e.g., will word embeddings be used directly in a traditional IR or machine learning model, or will they be input to an NN architecture for *end-to-end* learning? How should one deal with out-of-vocabulary (OOV) query terms not found in the word embedding training data for query-document matching?

Vulic and Moens (2015); Yang et al. (2016c); Zheng and Callan (2015); Zuccon et al. (2015) compare different training hyper-parameters such as window size and dimensionality of word embeddings. Zuccon et al. (2015)'s sensitivity analysis of the various model hyperparameters for inducing word embeddings shows that manipulations of embedding dimensionality, context window size, and model objective (CBOW vs skip-gram) have no consistent impact on model performance vs. baselines. Vulic and Moens (2015) find that while increasing dimensionality provides more semantic expressiveness, the impact on retrieval performance is relatively small. Zheng and Callan (2015) find that 100 dimensions work best for estimating term weights, better than 300 and 500. In experiments using Terrier, Yang et al. (2016c) find that for the Twitter election classification task using CNNs, word embeddings with a large context window and dimension size can achieve statistically significant improvements.

[21]`https://www.crowdflower.com`

Selection among `word2vec` CBOW or skip-gram or GloVe appears quite varied. Zuccon et al. (2015) compare CBOW vs. skip-gram, finding "no statistical significant differences between the two..." Kenter and de Rijke (2015) use both `word2vec` and GloVe (Pennington et al., 2014) embeddings (both originally released embeddings as well training their own embeddings) in order to induce features for their machine learning model. They report model effectiveness using the original public embeddings with or without their own additional embeddings, but do not report further ablation studies to understand the relative contribution of different embeddings used. Grbovic et al. (2015a)'s `query2vec` uses a two-level architecture in which the upper layer models the temporal context of query sequences via skip-gram, while the bottom layer models word sequences within a query using CBOW. However, these choices are not justified, and their later work (Grbovic et al., 2015b) uses skip-gram only. ALMasri et al. (2016) evaluate both CBOW and skip-gram `word2vec` embeddings (using default dimensionality and context window settings) but present only skip-gram results, writing that "there was no big difference in retrieval performance between the two." Zamani and Croft (2016b,a) adopt GloVe without explanation. Similarly for `word2vec`, Mitra et al. (2016) simply adopt CBOW, while De Vine et al. (2014); Manotumruksa et al. (2016); Vulic and Moens (2015); Yang et al. (2016b); Ye et al. (2016); Zhou et al. (2015) adopt skip-gram. Zhang and Wallace (2015) performed an empirical analysis in the context of using CNNs for short text classification. They found that the "best" embedding to use for initialization depended on the dataset. Motivated by this observation, the authors proposed a method for jointly exploiting multiple sets of embeddings (e.g., one embedding set induced using GloVe on some corpus and another using a `word2vec` variant on a different corpus) (Zhang et al., 2016b). This may also be fruitful for IR tasks, suggesting a potential direction for future work.

In selecting training data for word embeddings, Yang et al. (2016c) considers classification in which one background corpus (used to train word embeddings) is a Spanish Wikipedia Dump which contains over 1 million articles, while another is a collection of 20 million tweets having more than 10 words per tweet. As expected, they find that when the background training text matches the classification text, the performance is improved. On the other hand, Zuccon et al. (2015) also consider different training corpora, but find that "the choice of corpus used to construct word embeddings had little effect on retrieval results." Zamani and Croft (2016b) train GloVe on three external corpora and report, "there is no significant differences between the values obtained by employing different corpora for learning the embedding vectors." Regarding their model, Zheng and Callan (2015) write:

> "[the system] performed equally well with all three external corpora; the differences among them were too small and inconsistent to support any conclusion about which is best. However, although no external corpus was best for all datasets... The corpus-specific word vectors were never best in these experiments... given the wide range of training data sizes – varying from 250 million words to 100 billion words – it is striking how little correlation there is between search accuracy and the amount of training data."

Regarding handling of OOV terms, the easiest solution is to discard or ignore the OOV terms. For example, Zamani and Croft (2016b) only consider the queries where the embedding vectors of all terms are available. However, in end-to-end systems, where we are jointly estimating (or refining) embeddings alongside other model parameters, it is intuitive to randomly initialize embeddings for OOV words. For instance, in the context of CNNs for text classification, Kim (2014) adopted this approach. The intuition behind this is two-fold. First, if the same OOV appears in a pair of texts, queries, or documents being compared, this contributes to the similarity scores between those two. Second, if two different OOV terms appear in the same pair, their dissimilarity will not contribute in the similarity function. However, this does not specifically address accidental misspellings or social spellings (e.g., "kool") commonly found in social media. One might address this by hashing words to character n-grams (see Section 6.2) or character-based modeling more generally (e.g., Dhingra et al. (2016); Zhang et al. (2015), and Conneau et al. (2016)).

Particularly notable uses of embeddings include Mitra et al. (2016)'s exploiting of both the input and the output projections. They map query terms into the input space and the document terms into the output space. The motivation for this is that they can thus use both the embedding spaces to better capture the relatedness between queries and documents. Diaz et al. (2016)'s work is unique in developing topic-specific embeddings specific to a query (by training off a weighted sampled of its retrieved documents). They incorporate the new embedding into the original language model to get a better query language model for query expansion. Vulic and Moens (2015) learn bilingual word embeddings from comparable corpora rather than parallel corpora via a merge-and-shuffle approach, applying their model to cross-lingual IR.

Several authors train embeddings on query logs rather than documents (Mitra, 2015; Mitra and Craswell, 2015; Mitra et al., 2016; Sordoni et al., 2014, 2015). Grbovic et al. (2015b) train embeddings using `paragraph2vec` (Le and Mikolov, 2014) on the query and their session to obtain better query embeddings. They also propose the addition of ad clicks and search result clicks to the query context. This can help with query rewriting, which in turn can help retrieve more relevant ads.

Word embeddings have been learned from many other genres of text as well. Zhou et al. (2015) use the `word2vec` skip-gram model to train word embeddings on community questions and their metadata to get better embeddings for question terms. They constrain the words from the same category to be close to each other so that the trained word embedding can encode categorical information. Manotumruksa et al. (2016) use skip-gram to train word embeddings on venues' comments dataset from Foursquare to get better representation for venues and users' preferences, which they later use for venue recommendation. De Vine et al. (2014) use skip-gram to train on journal abstracts and patient records corpora to get embeddings specific for medical terms.

Similar to past development of new modeling techniques in IR, there is a common theme of researchers starting first with bag-of-words models then wanting to move toward modeling longer phrases in their future work. Ganguly et al. (2015) suggest future work should investigate compositionality of term embeddings. Zuccon et al. (2015) propose incorporating distributed representations of phrases to better model query term dependencies and compositionality. Zheng and Callan (2015) propose direct modeling of bigrams and proximity terms. Zamani and Croft (2016b) suggest query language models based on mutual-information and more complex language models (bigram, trigram, etc.) could be pursued. Kenter and de Rijke (2015) note that future work could extend their approach to consider word order.

In organizing this literature review, we have conceptually distinguished studies using word embeddings as inputs to an NN architecture (Section 5). In this case, the authors may further *fine-tune* word embeddings during NN training. For example, Sordoni et al. (2015) feeds embeddings into an RNN and then tunes embeddings during RNN training. On the other hand, Yang et al. (2016b) firstly *pre-train* word embeddings using the `word2vec` skip-gram model as the input to their neural network, then fix them during training. However, there is not yet any clear evidence showing that either method consistently works best.

6.2 Word Hashing

A number of papers (Shen et al., 2014b,a; Ye et al., 2015; Mitra, 2015; Mitra and Craswell, 2015) from Microsoft Research build on the DSSM model (Huang et al., 2013), likely due in part to DSSM's authors kindly sharing their learned model for others to use. DSSM is additionally notable in using word hashing to convert words into character n-grams, which are used as the input to neural network. Words are broken down into letter n-grams and then represented as a vector of letter n-grams. (Huang et al., 2013) provide empirical analysis where the original vocabulary size of $500k$ is reduced to only $30k$ because of word hashing. While the number of English words can be unlimited, the number of letter n-grams in English is often limited, thus word hashing can resolve the out-of-vocabulary (OOV) problem as well (Section 6.1). However, one inherent problem of word hashing is the hashing conflict which can be serious for a very large corpus. We note that this technique precedes the advent of Mikolov and Dean (2013); Mikolov et al. (2013))'s `word2vec`, and most studies since then use word embeddings rather than word hashing to encode words.

6.3 Representing Text Beyond Words

A simple way to represent longer textual units, such as phrases, sentences, or entire documents, is to simply sum or average their constituent word embeddings. However such bag-of-words composition ignores word ordering, and simple averaging treats all words as equally important in composition (though some work has considered weighted operations, e.g., Vulic and Moens (2015)). Ganguly et al. (2016) opine that:

> "...adding the constituent word vectors... to obtain the vector representation of the whole document is not likely to be useful, because... compositionality of the word vectors [only] works well when applied over a relatively small number of words... [and] does not scale well for a larger unit of text, such as passages or full documents, because of the broad context present within a whole document."

Fortunately, a variety of alternative methods have been proposed for inducing representations of such longer textual units. However, there is no evidence that any one method performs consistently best, with the performance of each method instead appearing to often depend on the specific task and dataset being studied. We further discuss such methods below.

Le and Mikolov (2014)'s oft-cited `ParagraphVector` (PV) trains a fixed-length vector representation for sentences and documents with two variants. The first proposed variant is Distributed Memory Model of PVs (PV-DM), which treats each paragraph as a unique token and concatenates/averages it with the word vectors in a context window in this paragraph. It then uses the concatenation/average to predict the following word in the context window. At test time, it needs to do inference on the unseen paragraph while fixing the word vectors already trained. The second proposed variant is Distributed Bag-of-Words (PV-DBOW), which uses PV to predict the word randomly sampled from the paragraph. Both variants train a fixed-length vector for sentences and documents, which can then be fed into a standard classifier like logistic regression. While `ParagraphVector` has been adopted in many studies (e.g., Xu and Cheng (2016) use PV-DBOW to represent documents) and is often reported as a baseline (e.g., (Tai et al., 2015)), concerns about reproducibility have also been raised. Kiros et al. (2015) report results below SVM when re-implementing `ParagraphVector`. Kenter and de Rijke (2015) note, "it is not clear, algorithmically, how the second step – the inference for new, unseen texts – should be carried out." Perhaps most significantly, later work co-authored by Mikolov (Mesnil et al., 2014)[22] has disavowed the original findings of Le and Mikolov (2014), writing, "to match the results from Le and Mikolov (2014), we followed the [author's] suggestion... However, this produces the [reported] result only when the training and test data are not shuffled. Thus, we consider this result to be invalid."

Balikas and Amini (2016) propose large-scale text classification using distributed document-level representations. The document-level representations are obtained from `word2vec` skip-gram word embeddings using naturally parallelizable composition functions based on neural networks. They further propose improving this method by combining document-level representations with traditional one-hot-encodings. Salakhutdinov and Hinton (2009) propose an auto-encoder approach to model documents. Gupta et al. (2014) also use auto-encoding to jointly model terms across mixed-scripts.

6.3.1 Convolutional Neural Networks (CNNs)

Shen et al. (2014a,b) propose a Convolutional Latent Semantic Model (CLSM) to encode queries and documents into fix-length vectors, following a popular "convolution+pooling" CNN architecture. The first layer of CLSM is a word hashing layer that can encode words into vectors. CLSM does not utilize word embeddings as input, seemingly distinguishing it from all other works using CNNs. Mitra (2015) use CLSM to encode query reformulations for query prediction, while Mitra and Craswell (2015) use CLSM on query prefix-suffix pairs corpus for query auto-completion. They sample queries from the search query logs and split the query at every possible word boundary to form prefix-suffix pairs.

Severyn and Moschitti (2015) and Suggu et al. (2016) adopt a similar "convolution+pooling" CNN architecture to encode question and answer sentence representations, which serve as features for a second-stage ranking NN. See Mitra et al. (2016) for further discussion of such two-stage *telescoping* approaches.

Yang et al. (2016b) develop a novel value-shared CNN, and apply it on the query-answer matching matrix to extract the semantic matching between the query and answer. This model can capture the

[22] `https://github.com/mesnilgr/iclr15`

interaction between intermediate terms in the query and answer, rather than only considering the final representation of the query and answer. The motivation behind the value-shared CNN is that semantic matching value regularities between a question and answer is more important than spatial regularities typical in computer vision. Similarly, contemporaneous work by Guo et al. (2016) notes:

> "Existing interaction-focused models, e.g., ARC-II and MatchPyramid, employ a CNN to learn hierarchical matching patterns over the matching matrix. These models are basically position-aware using convolutional units with a local "receptive field" and learning positional regularities in matching patterns. This may be suitable for the image recognition task, and work well on semantic matching problems due to the global matching requirement (i.e., all the positions are important). However, it may not be suitable for the ad-hoc retrieval task, since such positional regularity may not exist..."

Cohen et al. (2016) compare CNNs vs. RNNs based on document length (see Section 6.3.3).

6.3.2 Recurrent Neural Networks (RNNs)

Sordoni et al. (2015) build a hierarchical GRU to encode the query and the query session into vector representations. Song et al. (2016) use an LSTM to model user interests at different time steps and encode them into a vector. Lioma et al. (2016) create new relevant information using an LSTM that takes the concatenated text of a query and its known relevant documents as input using word embeddings. However, rather than take the whole text of a document, they extract a context window of $\pm n$ terms around every query term occurrence. Yan et al. (2016) use a bidirectional LSTM followed by a CNN to model the original query, reformulated query, candidate reply, and antecedent post in their human-computer conversation system. Wang and Nyberg (2015) use a stacked bidirectional LSTM to sequentially read words from both question and answer sentences, calculating relevance scores for answer sentence selection through mean pooling across all time steps. Section 6.3.3 presents Cohen et al. (2016)'s comparison of CNNs vs. RNNs by document length.

6.3.3 Text Granularity in IR

It is notable that many studies to date focus on short text matching rather than longer documents more typical of modern IR ad-hoc search. Cohen et al. (2016) study how the effectiveness of NN models for IR vary as a function of document length (i.e., text granularity). They consider three levels of granularity: i) *fine*, where documents often contain only a single sentence and relevant passages span only a few words (e.g., question answering); ii) *medium*, where documents consist of passages with a mean length of 75 characters and relevant information may span multiple sentences (e.g., passage retrieval); and iii) *coarse*, or typical modern ad-hoc retrieval. For fine granularity, they evaluate models using the TREC QA dataset and find that CNNs outperform RNNs and LSTM, as their filter lengths are able to effectively capture language dependencies. For medium granularity, they evaluate using the Yahoo Webscope L4 CQA dataset and conclude that LSTM networks outperform CNNs due to their ability to model syntactic and semantic dependencies independent of position in sequence. In contrast, RNNs without LSTM cells do not perform as well, as they tend to "forget" information due to passage length.

For ad-hoc retrieval performance on Robust04, comparing RNNs, CNNs, LSTM, DSSM, CLSM, Vulic and Moens (2015)'s approach, and Le and Mikolov (2014)'s `ParagraphVector`, Cohen et al. (2016) find that all neural models perform poorly. They note that neural methods often convert documents into fixed-length vectors, which can introduce a bias for either short or long documents. However, they find that the approaches of Vulic and Moens (2015) and Le and Mikolov (2014) perform well when combined with language modeling approaches which explicitly capture matching information of queries and documents.

6.4 Measuring Textual Similarity

Given a text representation, perhaps using word embeddings, how do we measure textual similarity? Similarity here is not strictly limited to linguistic semantics, but more generally includes relevance matching between queries and documents or questions and answers. While cosine similarity is the simplest and most common approach, a variety of more sophisticated methods have been proposed.

As noted at the start of Section 6.3, Ganguly et al. (2016) opine that simply adding vectors of word embedding cannot sufficiently capture context of longer textual units. Instead, the authors propose a new form of similarity metric based on the assumption that the document can be represented as a mixture of p-dimensional Gaussian probability density functions and each `word2vec` embedding (p-dimensions) is an observed sample. Then, using the EM algorithm, they estimate the probability density function which can be incorporated to the query likelihood language model using linear interpolation.

Zamani and Croft (2016a) and (Zamani and Croft, 2016b) propose using sigmoid and softmax transformations of cosine similarity on the grounds that cosine similarity values are not discriminative enough (Zamani and Croft, 2016a). Their empirical analysis shows that there are no substantial differences (e.g., two times more) between the similarity of the most similar term and the 1000th similar term to a given term w, while the 1000th word is unlikely to have any semantic similarity with w. Consequently, they propose using monotone mapping functions (e.g., sigmoid or softmax) to transform the cosine similarity scores.

Kenter and de Rijke (2015) propose a BM25 extension to incorporate word embeddings in order to calculate the similarity of short text. Their approach uses a word alignment method and a saliency-weighted semantic graph to move from word-level to text-level semantics. Features are computed from the word alignment method and from the means of word embeddings to train a final classifier that predicts a semantic similarity score.

Kusner et al. (2015) develop word movers distance (WMD) for computing distance between documents. It is observed by Mikolov and Dean (2013) that semantic relationships are often preserved in vector operations on `word2vec`. Thus, the proposed method (Kusner et al., 2015) utilizes this property of `word2vec` (see Section 3.1) and computes the minimum traveling distance from the embedded words of one document to another. This process is modeled as an Earth Mover's Distance (EMD) problem, a well studied optimization problem. Kim et al. (2016) propose another version of WMD specifically for query-document similarity measure. They handle the high computational cost of WMD by mapping queries to documents using a word embedding model trained on a document set. They make several changes to the original WMD methods: changing the weight of term by introducing inverse document frequency, and changing the original dissimilarity measure to cosine similarity. However, they do not provide any comparison vs. WMD as a baseline.

Zhang et al. (2014) use Fisher Kernel (Jaakkola et al., 1999) to calculate the similarity between short text segments given a trained word embedding to detect local text reuse. Clinchant and Perronnin (2013) and Zhou et al. (2015) also adopt a Fisher Kernel approach. Rekabsaz et al. (2016) develop a global similarity threshold to filter highly related terms having an expected number of synonyms for a word. Sordoni et al. (2014) embed queries and documents in a larger space than single terms on the grounds that text sequences have more informative content. Documents are scored for retrieval using a *quantum relative entropy* approach. Wang and Nyberg (2015) use a stacked bidirectional LSTM to sequentially read words from both question and answer sentences, calculating relevance scores for answer sentence selection through mean pooling across all time steps. Yan et al. (2016) match sentences by concatenating their vector representations and feeding them into a multi-layer fully-connected neural network, matching a query with the posting and reply in a human computer conversation system. Xu and Cheng (2016) propose a neural tensor network (NTN) approach to model document novelty. This model takes a document and a set of other documents as input. The architecture uses a tensor layer, a max-pooling layer, and a linear layer to output a document novelty score.

Guo et al. (2016)'s recent Deep Relevance Matching Model (DRMM) appears to be one of the most successful to date for ad-hoc search over longer document lengths. In terms of textual similarity, they argue that the ad-hoc retrieval task is mainly about relevance matching, different from semantic matching in NLP. They model the interaction between query terms and document terms, building a matching histogram on top of the similarities. They then feed the histogram into a feed forward neural network. They also use a term gating network to model the importance of each query term.

Yang et al. (2016b) propose an attention-based neural matching model (aNMM) for question answering. Similar to Guo et al. (2016), they first model the interaction between query terms and document terms to build a matching matrix. They then apply a novel value-shared CNN on the matrix. Since not all the query terms are equally important, they use a softmax gate function as an attention mechanism in order to learn the importance of each query term when calculating the matching between the query and the answer.

English Data	**Study**
20-Newsgroup Corpus	Salakhutdinov and Hinton (2009)
AOL Query Logs	Kanhabua et al. (2016), Mitra (2015), Mitra and Craswell (2015), Sordoni et al. (2015)
Bing Query Logs and WebCrawl	Mitra (2015), Mitra and Craswell (2015), Mitra et al. (2016), Nalisnick et al. (2016)
CLEF03 English Ad-hoc	Clinchant and Perronnin (2013)
CLEF Medical Corpora	ALMasri et al. (2016)
CLEF 2016 Social Book Search	Amer et al. (2016)
MSR Paraphrase Corpus	Kenter and de Rijke (2015)
MSN Query Log	Kanhabua et al. (2016)
OHSUMED	De Vine et al. (2014)
PubMed	Balikas and Amini (2016)
Reuters Volume I (RCV1-v2)	Salakhutdinov and Hinton (2009)
SemEval 2015-2016	DFFN (Suggu et al. (2016))
TIPSTER (Volume 1-3)	Zhang et al. (2014)
TREC 1-2 Ad-hoc (AP 88-89)	Clinchant and Perronnin (2013), Zamani and Croft (2016a), Zamani and Croft (2016b), Zuccon et al. (2015)
TREC 1-3 Ad-hoc	Zuccon et al. (2015)
TREC 6-8 Ad-hoc	GLM (Ganguly et al. (2015)), Lioma et al. (2016), Rekabsaz et al. (2016), Roy et al. (2016),
TREC 12 Ad-hoc	Diaz et al. (2016)
TREC 2005 HARD	Rekabsaz et al. (2016)
TREC 2007-2008 Million Query	Yang et al. (2016a)
TREC 2009-2011 Web	Xu and Cheng (2016)
TREC 2009-2013 Web	`ParagraphVector` (Grbovic et al. (2015b))
TREC 2010-2012 Web	QEM (Sordoni et al. (2014))
TREC 2011 Microblog	CDNN (Severyn and Moschitti (2015)), Zhang et al. (2014)
TREC 2012 Microblog	CDNN (Severyn and Moschitti (2015))
TREC 2015 Contextual Suggestion	Manotumruksa et al. (2016)
TREC ClueWeb09-Cat-B	DRMM (Guo et al. (2016)), QEM (Sordoni et al. (2014)), Zhang et al. (2014), Zheng and Callan (2015)
TREC DOTGOV	Zuccon et al. (2015)
TREC GOV2	Yang et al. (2016a), Zamani and Croft (2016a), Zamani and Croft (2016b), Zheng and Callan (2015)
TREC QA 8-13	aNMM (Yang et al. (2016b)), BLSTM (Wang and Nyberg (2015)), CDNN (Severyn and Moschitti (2015)), Yu et al. (2014), (Cohen et al., 2016)
TREC MedTrack	De Vine et al. (2014), Zuccon et al. (2015)
TREC Robust	GLM (Ganguly et al. (2015)), Roy et al. (2016)
TREC Robust 2004	Clinchant and Perronnin (2013), Diaz et al. (2016), DRMM (Guo et al. (2016)), Zamani and Croft (2016a), Zamani and Croft (2016b), Zheng and Callan (2015)
TREC WSJ87-92	Zuccon et al. (2015)

TREC WT10G	Roy et al. (2016), Zheng and Callan (2015)
Yahoo! Answers	Zhou et al. (2015)

Chinese Data	**Study**
Baidu Tieba	Yan et al. (2016)
Baidu Zhidao	Yan et al. (2016), Zhang et al. (2014), Zhou et al. (2015)
Douban Forum	Yan et al. (2016)
Sina Weibo	Yan et al. (2016), Zhang et al. (2014)
SogouT 2.0	Zhang et al. (2014)

Multi-Lingual Data	**Study**
CLEF 2001-2003 Ad-hoc	Vulic and Moens (2015)
FIRE 2013	Gupta et al. (2014)
iPinYou (Liao et al., 2014)	Zhang et al. (2016a)

Table 3: Datasets Used

System	**Citation**	**URL**
`word2vec`	Mikolov and Dean (2013)	`https://code.google.com/archive/p/word2vec/`
GloVe	Pennington et al. (2014)	`http://nlp.stanford.edu/projects/glove/`
CDNN	Severyn and Moschitti (2015)	`https://github.com/aseveryn/deep-qa`
DeepMerge	Lee et al. (2015)	`https://ciir.cs.umass.edu/downloads/DeepMerge/`
DeepTR	Zheng and Callan (2015)	`http://www.cs.cmu.edu/~gzheng/code/TermRecallKit-v2.tar.bz2`
Mixed Deep	Gupta et al. (2014)	`http://www.dsic.upv.es/~pgupta/mixed-script-ir`
NTLM	Zuccon et al. (2015)	`https://github.com/ielab/adcs2015-NTLM`

Table 4: Source Code Released

7 Conclusion

Interest in Neural IR has never been greater, spanning both active research and deployment in practice[23] (Metz, 2016). Neural IR continues to accelerate in quantity of work, sophistication of methods, and practical effectiveness (see Guo et al. (2016)). New methods are being explored that may be computationally infeasible today (see Diaz et al. (2016)), but if proven effective, could motivate future optimization work to make them more practically viable (e.g., (Jurgovsky et al., 2016; Ordentlich et al., 2016)). In his opening Keynote at SIGIR 2016, Chris Manning noted the rise of NN approaches to dominance in speech recognition (2011), computer vision (2013), and NLP (2015), and concluded with an unabashed assertion that, "I'm certain that deep learning will come to dominate SIGIR over the next couple of years... just like speech, vision, and NLP before it" (Manning, 2016).

[23]`https://en.wikipedia.org/wiki/RankBrain`

Data	Citation	URL
`word2vec` embeddings	Mikolov and Dean (2013)	`https://code.google.com/archive/p/word2vec/`
GloVe embeddings	Pennington et al. (2014)	`http://nlp.stanford.edu/projects/glove/`
Bing query embeddings	Mitra et al. (2016)	`https://www.microsoft.com/en-us/download/details.aspx?id=52597`
NTLM embeddings	Zuccon et al. (2015)	`http://www.zuccon.net/ntlm.html`

Table 5: Datasets Released

At the same time, healthy skepticism about Neural IR also remains. Despite his bold assertion, Manning nevertheless reminded his audience to always be wary of the common "emerging technology hype cycle" of having unrealistic expectations early in the development of any promising new technology. The key question in IR today might be most succinctly expressed as: "Will it work?". While NN methods have worked quite well on short texts, effectiveness on longer texts typical of ad-hoc search has been problematic (Huang et al., 2013; Cohen et al., 2016), with only very recent evidence to the contrary (Guo et al., 2016). In addition, while great strides have been made in computer vision through employing a very large number of *hidden layers* (hence "deep" learning), such deep structures have typically been less effective in NLP and IR than shallower architectures (Pang et al., 2016a), though again with notable recent exceptions (see (Conneau et al., 2016)). When Neural IR has improved in ad-hoc search results, improvements appear relatively modest (Zamani and Croft, 2016a; Diaz et al., 2016) when compared to traditional query expansion techniques for addressing *vocabulary mismatch*, such as pseudo-relevance feedback (PRF). Both Ganguly et al. (2016) and Diaz et al. (2016) have noted that *global* word embeddings, trained without reference to user queries, vs. *local* methods like PRF for exploiting query-context, appear limited similarly to the traditional global-local divide seen with existing approaches like topic modeling (Yi and Allan, 2009).

As Li (2016a) so eloquently put it, "Does IR Need Deep Learning?" Such a seemingly simple question requires careful unpacking. Much of the above discussion assumes Neural IR should deliver new state-of-the-art quality of search results for traditional search tasks. While it may do so, this framing may be far too narrow, as Li (2016a)'s presentation suggests. The great strength of Neural IR may lie in enabling a new generation of search scenarios and modalities, such as searching via conversational agents (Yan et al., 2016), multi-modal retrieval (Ma et al., 2015c,a), knowledge-based search IR (Nguyen et al., 2016), or synthesis of relevant material (Lioma et al., 2016). It may also be that Neural IR will provide greater traction for other future search scenarios not yet considered.

Given that efficacy of deep learning approaches is often driven by "big data", will Neural IR represent yet another fork in the road between industry and academic research, where massive commercial query logs deliver Neural IR's true potential? There is also an important contrast to note here between supervised scenarios, such as learning to rank (Liu, 2009) vs. unsupervised learning of word embeddings or typical queries (see Mitra and Craswell (2015); Mitra (2015); Sordoni et al. (2015)). LeCun et al. (2015) wrote, "we expect unsupervised learning to become far more important in the longer term." Just as the rise of the Web drove work on unsupervised and semi-supervised approaches by the sheer volume of unlabeled data it made available, the greatest value of Neural IR may naturally arise where the biggest data is found: continually generated and ever-growing behavioral traces in search logs, as well as ever-growing online content.

While skepticism of Neural IR may well remain for some time, the practical importance of search today, coupled with the potential for significantly new traction offered by this "third wave" of NNs, makes it unlikely that researchers will abandon Neural IR anytime soon without having first exhaustively tested its limits. As such, we expect the pace and interest in Neural IR will only continue to blossom, both in new research and increasing application in practice.

8 Additional Authors

The following additional students at the University of Texas at Austin contributed indirectly to the writing of this literature review: Manu Agarwal, Edward Babbe, Anuparna Banerjee, Jason Cai, Dillon Caryl, Yung-sheng Chang, Shobhit Chaurasia, Linli Ding, Brian Eggert, Michael Feilbach, Alan Gee, Jeremy Gin, Rahul Huilgol, Miles Hutson, Neha Javalagi, Yan Jiang, Kunal Lad, Yang Liu, Amanda Lucio, Kristen Moor, Daniel Nelson, Geoffrey Potter, Harshal Priyadarshi, Vedhapriya Raman, Eric Roquemore, Juliette Seive, Abhishek Sinha, Ashwini Venkatesh, Yuxuan Wang, and Xu Zhang.

References

Abdul-Jaleel, N., Allan, J., Croft, W. B., Diaz, F., Larkey, L., Li, X., Smucker, M. D., and Wade, C. (2004). Umass at trec 2004: Novelty and hard.

ALMasri, M., Berrut, C., and Chevallet, J.-P. (2016). A Comparison of Deep Learning Based Query Expansion with Pseudo-Relevance Feedback and Mutual Information. In *European Conference on Information Retrieval*, pages 709–715. Springer.

Amati, G. and Van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389.

Amer, N. O., Mulhem, P., and Gery, M. (2016). Toward Word Embedding for Personalized Information Retrieval. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.06991.

Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468. IEEE.

Arel, I., Rose, D. C., and Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, 5(4):13–18.

Azzopardi, L., Girolami, M., and Crowe, M. (2005). Probabilistic hyperspace analogue to language. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 575–576. ACM.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Chapelle, O., and Weinberger, K. (2009). Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 187–196. ACM.

Balikas, G. and Amini, M.-R. (2016). An empirical study on large scale text classification with skip-gram embeddings. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.06623.

Bar-Yossef, Z. and Kraus, N. (2011). Context-sensitive query auto-completion. In *Proceedings of the 20th international conference on World wide web*, pages 107–116. ACM.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127.

Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–229. ACM.

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620. Association for Computational Linguistics.

Braschler, M. and Peters, C. (2004). Cross-language evaluation forum: Objectives, results, achievements. *Information retrieval*, 7(1-2):7–31.

Broder, A., Domingos, P., de Freitas, N., Guyon, I., Malik, J., and Neville, J. (2016). Is deep learning the new 42? In *Plenary Panel at the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

Bruza, P. D. and Song, D. (2002). Inferring query models by computing information flow. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 260–269. ACM.

Cai, F. and de Rijke, M. (2016). Learning from homologous queries and semantically related terms for query auto completion. *Information Processing & Management*, 52(4):628–643.

Cai, F., Liang, S., and De Rijke, M. (2014). Time-sensitive personalized query auto-completion. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1599–1608. ACM.

Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'El, N., Ronen, I., Uziel, E., Yogev, S., and Chernov, S. (2009). Personalized social search based on the user's social network. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1227–1236. ACM.

Cartright, M.-A., Allan, J., Lavrenko, V., and McGregor, A. (2010). Fast query expansion using approximations of relevance models. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1573–1576. ACM.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM.

Chirita, P.-A., Firan, C. S., and Nejdl, W. (2007). Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 7–14. ACM.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning*. arXiv:1412.3555.

Clinchant, S. and Perronnin, F. (2013). Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109.

Cohen, D., Ai, Q., and Croft, W. B. (2016). Adaptability of Neural Networks on Varying Granularity IR Tasks. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07565.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2016). Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Croft, B., Metzler, D., and Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company.

Dang, V. and Croft, B. W. (2010). Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 41–50. ACM.

Darragh, J. J., Witten, I. H., and James, M. L. (1990). The reactive keyboard: A predictive typing aid. *Computer*, 23(11):41–49.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM.

De Vine, L., Zuccon, G., Koopman, B., Sitbon, L., and Bruza, P. (2014). Medical semantic similarity with a neural language model. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1819–1822. ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2.

Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., and Cohen, W. (2016). Tweet2vec: Character-based distributed representations for social media. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 269–274. Association for Computational Linguistics.

Diaz, F., Mitra, B., and Craswell, N. (2016). Query Expansion with Locally-Trained Word Embeddings. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 367–377, Berlin, Germany. Association for Computational Linguistics. Earlier version appeared at SIGIR 2016 Neuro-IR workshop.

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. (2002). Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298. ACM.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2014). Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis, Philological Society*. Reprinted in Palmer, F. (ed.) 1968 Selected Papers of J. R. Firth, Longman, Harlow.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Ganguly, D., Roy, D., Mitra, M., and Jones, G. (2016). Representing Documents and Queries as Sets of Word Embedded Vectors for Information Retrieval. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07869.

Ganguly, D., Roy, D., Mitra, M., and Jones, G. J. (2015). Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 795–798. ACM.

Gao, J. (2015). Deep Learning for Web Search and Natural Language Processing.

Gao, J., He, X., and Nie, J.-Y. (2010). Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1139–1148. ACM.

Gao, J., Pantel, P., Gamon, M., He, X., and Deng, L. (2014). Modeling Interestingness with Deep Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2–13, Doha, Qatar. Association for Computational Linguistics.

Glowacka, D., Ruotsalo, T., Konuyshkova, K., Kaski, S., Jacucci, G., et al. (2013). Directing exploratory search: Reinforcement learning from user interactions with keywords. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 117–128. ACM.

Goldberg, Y. (2015). A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. Book in preparation for MIT Press.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Grbovic, M., Djuric, N., Radosavljevic, V., and Bhamidipati, N. (2015a). Search retargeting using directed query embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 37–38. ACM.

Grbovic, M., Djuric, N., Radosavljevic, V., Silvestri, F., and Bhamidipati, N. (2015b). Context-and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 383–392. ACM.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A Deep Relevance Matching Model for Ad-hoc Retrieval. In *The 25th ACM International Conference on Information and Knowledge Management*, Indianapolis, United States.

Gupta, P., Bali, K., Banchs, R. E., Choudhury, M., and Rosso, P. (2014). Query Expansion for Mixed-Script Information Retrieval.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and others (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Jaakkola, T. S., Haussler, D., et al. (1999). Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493.

Jagarlamudi, J., Udupa, R., Daumé III, H., and Bhole, A. (2011). Improving bilingual projections via sparse covariance matrices. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 930–940. Association for Computational Linguistics.

Jiang, J.-Y., Ke, Y.-Y., Chien, P.-Y., and Cheng, P.-J. (2014). Learning user reformulation behavior for query auto-completion. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 445–454. ACM.

Jurgovsky, J., Granitzer, M., and Seifert, C. (2016). Evaluating Memory Efficiency and Robustness of Word Embeddings. In *European Conference on Information Retrieval*, pages 200–211. Springer.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kanhabua, N., Ren, H., and Moeslund, T. B. (2016). Learning Dynamic Classes of Events using Stacked Multilayer Perceptron Networks. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07219.

Kenter, T. and de Rijke, M. (2015). Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM.

Kiela, D. and Clark, S. (2013). Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *EMNLP*, pages 1427–1432.

Kim, S., Wilbur, W. J., and Lu, Z. (2016). Bridging the Gap: a Semantic Similarity Measure between Queries and Documents. *arXiv preprint arXiv:1608.01972*.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Kusner, M. J., Sun, Y., Kolkin, N. I., and Weinberger, K. Q. (2015). From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966.

Lalmas, M. and Tombros, A. (2007). Evaluating xml retrieval effectiveness at inex. In *ACM SIGIR Forum*, volume 41, pages 40–57. ACM.

Lavrenko, V. and Croft, W. B. (2001). Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM.

Le, Q. V. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. In *ICML*, volume 14, pages 1188–1196.

Lease, M., Allan, J., and Croft, W. B. (2009). Regression rank: Learning to meet the opportunity of descriptive queries. In *European Conference on Information Retrieval*, pages 90–101. Springer.

LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, C.-J., Ai, Q., Croft, W. B., and Sheldon, D. (2015). An Optimization Framework for Merging Multiple Result Lists. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 303–312. ACM.

Li, H. (2016a). Does IR Need Deep Learning?

Li, H. (2016b). Opportunities and Challenges in Deep Learning for Information Retrieval.

Li, H. and Lu, Z. (2016). Deep Learning for Information Retrieval. In *SIGIR*, volume Tutorial at the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 1203–1206, Pisa, Italy. Slides online at http://www.hangli-hl.com/uploads/3/4/4/6/34465961/deep_learning_for_information_retrieval.pdf.

Li, X., Guo, C., Chu, W., Wang, Y.-Y., and Shavlik, J. (2014). Deep learning powered in-session contextual ranking using clickthrough data. In *In Proc. of NIPS*.

Liao, H., Peng, L., Liu, Z., and Shen, X. (2014). ipinyou global rtb bidding algorithm competition dataset. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–6. ACM.

Lioma, C., Larsen, B., Petersen, C., and Simonsen, J. G. (2016). Deep Learning Relevance: Creating Relevant Information (as Opposed to Retrieving it). In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv preprint arXiv:1606.07660.

Lioma, C., Simonsen, J. G., Larsen, B., and Hansen, N. D. (2015). Non-Compositional Term Dependence for Information Retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–604. ACM.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Liu, W., Wang, J., Ji, R., Jiang, Y.-G., and Chang, S.-F. (2012). Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE.

Liu, X. and Croft, W. B. (2004). Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM.

Lu, Z. and Li, H. (2013). A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375.

Lund, K. and Burgess, C. (1996). Hyperspace analogue to language (hal): A general model semantic representation. In *Brain and Cognition*, volume 30, pages 5–5. ACADEMIC PRESS INC JNL-COMP SUBSCRIPTIONS 525 B ST, STE 1900, SAN DIEGO, CA 92101-4495.

Luukkonen, P., Koskela, M., and Floreen, P. (2016). LSTM-Based Predictions for Proactive Information Retrieval. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.06137.

Ma, L., Lu, Z., and Li, H. (2015a). Learning to answer questions from image using convolutional neural network. *arXiv preprint arXiv:1506.00333*.

Ma, L., Lu, Z., and Li, H. (2016). Learning to answer questions from image using convolutional neural network. In *AAAI Conference on Artificial Intelligence*, pages 3567–3573.

Ma, L., Lu, Z., Shang, L., and Li, H. (2015b). Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631.

Ma, L., Lu, Z., Shang, L., and Li, H. (2015c). Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631.

Manning, C. (2016). Natural Language Inference, Reading Comprehension and Deep Learning.

Manning, C. D., Raghavan, P., and Schtze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Manotumruksa, J., Macdonald, C., and Ounis, I. (2016). Modelling User Preferences using Word Embeddings for Context-Aware Venue Recommendation. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07828.

Mesnil, G., He, X., Deng, L., and Bengio, Y. (2013). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH*, pages 3771–3775.

Mesnil, G., Mikolov, T., Ranzato, M., and Bengio, Y. (2014). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. In *International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1412.5335.

Metz, C. (2016). Ai is transforming google search. the rest of the web is next. *WIRED Magazine*.

Metzler, D. and Croft, W. B. (2005). A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

Mitra, B. (2015). Exploring session context using distributed representations of queries and reformulations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM.

Mitra, B. and Craswell, N. (2015). Query auto-completion for rare prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1755–1758. ACM. Issues: Future Work Dateset description Please add relevant tag.

Mitra, B., Nalisnick, E., Craswell, N., and Caruana, R. (2016). A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*. Extends WWW 2016 poster: Improving document ranking with dual word embeddings.

Moshfeghi, Y., Triantafillou, P., and Pollick, F. E. (2016). Understanding information need: An fmri study. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 335–344, New York, NY, USA. ACM.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Nalisnick, E., Mitra, B., Craswell, N., and Caruana, R. (2016). Improving document ranking with dual word embeddings. In *25th World Wide Web (WWW) Conference Companion Volume*, pages 83–84. International World Wide Web Conferences Steering Committee. See also extended arXiv paper: A dual embedding space model for document ranking. \url{http://arxiv.org/abs/1602.01137}.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069.

Nguyen, G.-H., Tamine, L., Soulier, L., and Bricon-Souf, N. (2016). Toward a Deep Neural Approach for Knowledge-Based IR. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:xxx.

Ordentlich, E., Yang, L., Feng, A., Cnudde, P., Grbovic, M., Djuric, N., Radosavljevic, V., and Owens, G. (2016). Network-Efficient Distributed Word2vec Training System for Large Vocabularies. In *The 25th ACM International Conference on Information and Knowledge Management*, Indianapolis, United States.

Pang, L., Lan, Y., Guo, J., Xu, J., and Cheng, X. (2016a). A Study of MatchPyramid Models on Ad-hoc Retrieval. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.04648.

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., and Cheng, X. (2016b). Text Matching as Image Recognition. In *30th AAAI Conference on Artificial Intelligence*, pages 2793–2799.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Ponte, J. M. and Croft, W. B. (1998). A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM.

Rasolofo, Y. and Savoy, J. (2003). Term proximity scoring for keyword-based retrieval systems. In *European Conference on Information Retrieval*, pages 207–218. Springer.

Rekabsaz, N., Lupu, M., and Hanbury, A. (2016). Uncertainty in Neural Network Word Embedding Exploration of Potential Threshold. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.06086.

Rish, I. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York.

Rocchio, J. J. (1971). Relevance feedback in information retrieval.

Roy, D., Paul, D., and Mitra, M. (2016). Using Word Embeddings for Automatic Query Expansion. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07608.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Salakhutdinov, R. and Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978. Extends earlier version presented at the SIGIR 2007 Workshop: Information Retrieval and Applications of Graphical Models.

Sanderson, M. (2010). *Test collection based evaluation of information retrieval systems*. Now Publishers Inc.

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117. arXiv: 1404.7828.

Severyn, A. and Moschitti, A. (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. ACM.

Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014a). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM.

Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014b). Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM.

Shokouhi, M. and Radinsky, K. (2012). Time-sensitive query auto-completion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 601–610. ACM.

Singhal, A. and Pereira, F. (1999). Document expansion for speech retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 34–41. ACM.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document.

Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Song, Y., Elkahky, A. M., and He, X. (2016). Multi-Rate Deep Learning for Temporal Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 909–912, New York, NY, USA. ACM.

Sordoni, A., Bengio, Y., and Nie, J.-Y. (2014). Learning Concept Embeddings for Query Expansion by Quantum Entropy Minimization. In *AAAI*, pages 1586–1592.

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Grue Simonsen, J., and Nie, J.-Y. (2015). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 553–562. ACM.

Suggu, S. P., Goutham, K. N., Chinnakotla, M. K., and Shrivastava, M. (2016). Deep Feature Fusion Network for Answer Quality Prediction in Community Question Answering. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07103.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Tao, T. and Zhai, C. (2007). An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 295–302. ACM.

Voorhees, E. M. and Harman, D. K. (2005). *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press.

Vulic, I. and Moens, M.-F. (2015). Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 363–372. ACM.

Wan, J., Wang, D., Hoi, S. C. H., Wu, P., Zhu, J., Zhang, Y., and Li, J. (2014). Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. ACM.

Wang, D. and Nyberg, E. (2015). A long short-term memory model for answer sentence selection in question answering. *ACL, July*.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32.

Wei, X. and Croft, W. B. (2006). Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185. ACM.

Weiss, Y., Torralba, A., and Fergus, R. (2009). Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760.

Weston, J., Bengio, S., and Usunier, N. (2010). Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. In *International Conference on Learning Representations (ICLR)*. arXiv preprint arXiv:1410.3916.

Wu, Q., Burges, C. J., Svore, K. M., and Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xia, L., Xu, J., Lan, Y., Guo, J., and Cheng, X. (2015). Learning maximal marginal relevance model via directly optimizing diversity evaluation measures. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–122. ACM.

Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., and Liu, T.-Y. (2014). Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

Xu, L. X. J. and Cheng, Y. L. J. G. X. (2016). Modeling Document Novelty with Neural Tensor Network for Search Result Diversification. In *SIGIR*, Pisa, Italy.

Yan, R., Song, Y., and Wu, H. (2016). Learning to Respond with Deep Neural Networks for Retrieval-Based Human-Computer Conversation System. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Yang, J., Stones, R., Wang, G., and Liu, X. (2016a). Selective Term Proximity Scoring Via BP-ANN. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07188.

Yang, L., Ai, Q., Guo, J., and Croft, W. B. (2016b). aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In *The 25th ACM International Conference on Information and Knowledge Management*, Indianapolis, United States.

Yang, X., Macdonald, C., and Ounis, I. (2016c). Using Word Embeddings in Twitter Election Classification. In *ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR)*. arXiv:1606.07006.

Ye, X., Qi, Z., and Massey, D. (2015). Learning relevance from click data via neural network based similarity models. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 801–806. IEEE.

Ye, X., Shen, H., Ma, X., Bunescu, R., and Liu, C. (2016). From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th International Conference on Software Engineering*, pages 404–415. ACM.

Yi, X. and Allan, J. (2009). A comparative study of utilizing topic models for information retrieval. In *European Conference on Information Retrieval*, pages 29–41. Springer.

Yu, D. and Deng, L. (2011). Deep Learning and Its Applications to Signal and Information Processing [Exploratory DSP]. *IEEE Signal Processing Magazine*, 28(1):145–154.

Yu, L., Hermann, K. M., Blunsom, P., and Pulman, S. (2014). Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Zamani, H. and Croft, W. B. (2016a). Embedding-based Query Language Models. In *2nd ACM International Conference on the Theory of Information Retrieval*, page TBD.

Zamani, H. and Croft, W. B. (2016b). Estimating Embedding Vectors for Queries. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 123–132. ACM.

Zhang, Q., Kang, J., Qian, J., and Huang, X. (2014). Continuous word embeddings for detecting local text reuses at the semantic level. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 797–806. ACM.

Zhang, W., Du, T., and Wang, J. (2016a). Deep Learning over Multi-field Categorical Data. In *European Conference on Information Retrieval*, pages 45–57. Springer.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Zhang, Y., Roller, S., and Wallace, B. C. (2016b). MGNC-CNN: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527, San Diego, California. Association for Computational Linguistics.

Zhang, Y. and Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Zhao, L. and Callan, J. (2010). Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 259–268. ACM.

Zheng, G. and Callan, J. (2015). Learning to Reweight Terms with Distributed Representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 575–584, New York, NY, USA. ACM.

Zhou, G., He, T., Zhao, J., and Hu, P. (2015). Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, pages 250–259.

Zhu, Y., Lan, Y., Guo, J., Cheng, X., and Niu, S. (2014). Learning for search result diversification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 293–302. ACM.

Zuccon, G., Koopman, B., Bruza, P., and Azzopardi, L. (2015). Integrating and Evaluating Neural Word Embeddings in Information Retrieval. In *Proceedings of the 20th Australasian Document Computing Symposium*, page 12. ACM. Slides at `https://github.com/ielab/adcs2015-NTLM/blob/master/adcs2015presentation/adcs2015_ntlm.pdf`.