An Introduction to Neural Information Retrieval

Bhaskar Mitra[1] and Nick Craswell[2]

[1] *Microsoft, University College London; bmitra@microsoft.com*
[2] *Microsoft; nickcr@microsoft.com*

ABSTRACT

Neural ranking models for information retrieval (IR) use shallow or deep neural networks to rank search results in response to a query. Traditional learning to rank models employ supervised machine learning (ML) techniques—including neural networks—over hand-crafted IR features. By contrast, more recently proposed neural models learn representations of language from raw text that can bridge the gap between query and document vocabulary. Unlike classical learning to rank models and non-neural approaches to IR, these new ML techniques are data-hungry, requiring large scale training data before they can be deployed. This tutorial introduces basic concepts and intuitions behind neural IR models, and places them in the context of classical non-neural approaches to IR. We begin by introducing fundamental concepts of retrieval and different neural and non-neural approaches to unsupervised learning of vector representations of text. We then review IR methods that employ these pre-trained neural vector representations without learning the IR task end-to-end. We introduce the Learning to Rank (LTR) framework next, discussing standard loss functions for ranking. We follow that with an overview of deep neural networks (DNNs), including standard architectures and implementations. Finally, we review supervised neural learning to rank models, including recent DNN architectures trained end-to-end for ranking tasks. We conclude with a discussion on potential future directions for neural IR.

1

Introduction

Since the turn of the decade, there have been dramatic improvements in performance in computer vision, speech recognition, and machine translation tasks, witnessed in research and in real-world applications (LeCun *et al.*, 2015). These breakthroughs were largely fuelled by recent advances in neural network models, usually with multiple hidden layers, known as deep architectures (Krizhevsky *et al.*, 2012; LeCun *et al.*, 2015; Hinton *et al.*, 2012; Bahdanau *et al.*, 2014; Deng, Yu, *et al.*, 2014) combined with the availability of large datasets (Wissner-Gross, 2016) and cheap compute power for model training. Exciting novel applications, such as conversational agents (Vinyals and Le, 2015; Sordoni *et al.*, 2015b), have also emerged, as well as game-playing agents with human-level performance (Silver *et al.*, 2016; Mnih *et al.*, 2015). Work has now begun in the information retrieval (IR) community to apply these neural methods, leading to the possibility of advancing the state of the art or even achieving breakthrough performance as in these other fields.

Retrieval of information can take many forms (White, 2016). Users can express their information need in the form of a text query—by typing on a keyboard, by selecting a query suggestion, or by voice recognition—or the query can be in the form of an image, or in some

cases the need can be implicit. Retrieval can involve ranking existing pieces of content, such as documents or short-text answers, or composing new responses incorporating retrieved information. Both the information need and the retrieved results may use the same modality (*e.g.*, retrieving text documents in response to keyword queries), or be different (*e.g.*, image search using text queries). If the query is ambiguous, retrieval system may consider user history, physical location, temporal changes in information, or other context when ranking results. IR systems may also help users formulate their intent (*e.g.*, via query auto-completion or query suggestion) and can extract succinct summaries of results that take the user's query into account.

We note that many natural language processing tasks exist that are not IR. Machine translation of text from one human language to another is not an IR task, because translating language and searching a corpus to satisfy a user's information need are different. However, translation could be used in an IR system, to enable cross-language retrieval on a multilingual corpus (Oard and Diekema, 1998). Named entity linking, where text is disambiguated through linking to a knowledgebase, is not an IR task in itself. However, an IR system could use entity linking to enhance its performance on IR tasks. In general, many natural language processing tasks do not involve information access and retrieval, so are not IR tasks, but some can still be useful as part of a larger IR system.

✎ Neural IR is the application of shallow or deep neural networks to IR tasks. Other natural language processing capabilities such as machine translation and named entity linking are *not* neural IR but could be used in an IR system.

Neural IR refers to the application of shallow or deep neural networks to retrieval tasks. Neural models have been employed in many

IR scenarios—including ad-hoc retrieval, recommender systems, multimedia search, and even conversational systems that generate answers in response to natural language questions. This tutorial serves as an introduction to neural methods for ranking documents in response to a query, an important IR task. We scope our discussions to a single task to allow for more thorough treatment of the fundamentals as opposed to providing a shallow survey of neural approaches to all IR tasks.

A search query may typically contain a few terms, while the document length, depending on the scenario, may range from a few terms to hundreds of sentences or more. Neural models for IR use vector representations of text, and usually contain a large number of parameters that need to be tuned. ML models with large set of parameters typically benefit from large quantity of training data (Brill, 2003; Taylor *et al.*, 2006; Rajaraman, 2008; Halevy *et al.*, 2009; Sun *et al.*, 2017). Unlike traditional *learning to rank* (LTR) approaches (Liu, 2009) that train ML models over a set of hand-crafted features, recent neural models for IR typically accept the raw text of a query and document as input. Learning suitable representations of text also demands large-scale datasets for training (Mitra *et al.*, 2017a). Therefore, unlike classical IR models, these neural approaches tend to be data hungry, with performance that improves with more training data.

Text representations can be learnt in an unsupervised or supervised fashion. The supervised approach uses IR data such as labelled query-document pairs, to learn a representation that is optimized end-to-end for the task at hand. If sufficient IR labels are not available, the unsupervised approach learns a representation using just the queries and/or documents. In the latter case, different unsupervised learning setups may lead to vector representations that capture different notions of text similarity. When applying such representations, the choice of unsupervised learning setup should be carefully considered, to yield a notion of text similarity that is suitable for the target task. Traditional IR models such as *Latent Semantic Analysis* (LSA) (Deerwester *et al.*, 1990) learn dense vector representations of terms and documents. Neural representation learning models share commonalities with these traditional approaches. Much of our understanding of these traditional
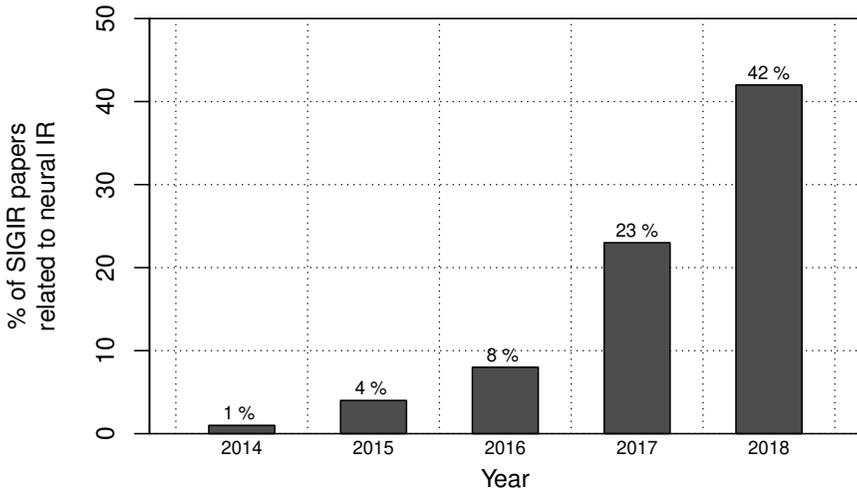
Figure 1.1: The percentage of neural IR papers at the ACM SIGIR conference—as determined by a manual inspection of the papers—shows a clear trend in the growing popularity of the field.

approaches from decades of research can be extended to these modern representation learning models.

In other fields, the design of neural network models has been informed by characteristics of the application and data. For example, the datasets and successful architectures are quite different in visual object recognition, speech recognition, and game playing agents. While IR shares some common attributes with the field of natural language processing, it also comes with its own set of unique challenges. IR systems must deal with short queries that may contain previously unseen vocabulary, to match against documents that vary in length, to find relevant documents that may also contain large sections of irrelevant text. IR systems should learn patterns in query and document text that indicate relevance, even if query and document use different vocabulary, and even if the patterns are task-specific or context-specific.

The goal of this tutorial is to introduce the fundamentals of neural IR, in context of traditional IR research, with visual examples to illustrate key concepts and a consistent mathematical notation for describing key models. Section 2 presents a survey of IR tasks, challenges, metrics

and non-neural models—as well as a brief overview of different neural approaches to IR. Section 3 introduces neural and non-neural methods for learning term embeddings, without the use of supervision from IR labels, and with a focus on the notions of similarity. Section 4 surveys some specific approaches for incorporating such unsupervised embeddings in IR. Section 5 introduces supervised learning to rank models. Section 6 introduces the fundamentals of deep models—including standard architectures and toolkits—before §7 surveys some specific approaches for incorporating deep neural networks (DNNs) in IR. Section 8 is our discussion, including future work, and conclusion.

Motivation for this tutorial   Neural IR is an emerging field. Research publication in the area has been increasing (Figure 1.1), along with relevant workshops (Craswell *et al.*, 2016a; Craswell *et al.*, 2016b; Craswell *et al.*, 2017; Craswell *et al.*, 2018), tutorials (Li and Lu, n.d.; Mitra and Craswell, 2017; Kenter *et al.*, 2017; Kenter *et al.*, 2018a; Kenter *et al.*, 2018b), and plenary talks (Manning, 2016; Craswell, 2017). Because this growth in interest is fairly recent, some researchers with IR expertise may be unfamiliar with neural models, and other researchers who have already worked with neural models may be unfamiliar with IR. The purpose of this tutorial is to bridge the gap, by describing the relevant IR concepts and neural methods in the current literature.

A thorough review of the fundamentals of IR or neural networks, however, are beyond the scope of this tutorial. We refer interested readers to the following books for a general overview of traditional IR.

1. *Modern information retrieval*, by Baeza-Yates and Ribeiro-Neto (1999).

2. *Introduction to information retrieval*, by Manning *et al.* (2008).

3. *Search engines: Information retrieval in practice*, by Croft *et al.* (2010).

Similarly, we recommend the following as companion reading materials for machine learning and neural network fundamentals.

1. *The elements of statistical learning*, by Hastie *et al.* (2001).

2. *Pattern recognition and machine learning*, by Bishop (2006).

3. *Deep learning*, by Goodfellow *et al.* (2016).

2

Fundamentals of text retrieval

We focus on text retrieval in IR, where the user enters a text query and the system returns a ranked list of search results. Search results may be passages of text or full text documents. The system's goal is to rank the user's preferred search results at the top. This problem is a central one in the IR literature, with well-understood challenges and solutions. This section provides an overview of those, such that we can refer to them in subsequent sections.

2.1 IR tasks

Text retrieval methods for full text documents and for short text passages have application in ad hoc retrieval systems and question answering systems, respectively.

Ad-hoc retrieval Ranked document retrieval is a classic problem in information retrieval, as in the main task of the Text Retrieval Conference (Voorhees, Harman, *et al.*, 2005), and performed by commercial search engines such as Google, Bing, Baidu, or Yandex. TREC tasks may offer a choice of query length, ranging from a few terms to a few sentences, whereas search engine queries tend to be at the shorter end

of the range. In an operational search engine, the retrieval system uses specialized index structures to search potentially billions of documents. The results ranking is presented in a search engine results page (SERP), with each result appearing as a summary and a hyperlink. The engine can instrument the SERP, gathering implicit feedback on the quality of search results such as click decisions and dwell times.

A ranking model can take a variety of input features. Some ranking features may depend on the document alone, such as how popular the document is with users, how many incoming links it has, or to what extent document seems problematic according to a Web spam classifier. Other features depend on how the query matches the text content of the document. Still more features match the query against document metadata, such as referred text of incoming hyperlink anchors, or the text of queries from previous users that led to clicks on this document. Because anchors and click queries are a succinct description of the document, they can be a useful source of ranking evidence, but they are not always available. A newly created document would not have much link or click text. Also, not every document is popular enough to have past links and clicks, but it still may be the best search result for a user's rare or tail query. In such cases, when text metadata is unavailable, it is crucial to estimate the document's relevance primarily based on its text content.

In the text retrieval community, retrieving documents for short-text queries by considering the long body text of the document is an important challenge. These *ad-hoc* retrieval tasks have been an important part of the Text REtrieval Conference (TREC) (Voorhees and Harman, 2000), starting with the original tasks searching newswire and government documents, and later with the Web track[1] among others. The TREC participants are provided a set of, say fifty, search queries and a document collection containing 500-700K newswire and other documents. Top ranked documents retrieved for each query from the collection by different competing retrieval systems are assessed by human annotators based on their relevance to the query. Given a query, the goal of the IR model is to rank documents with better assessor

---

[1] http://www10.wwwconference.org/cdrom/papers/317/node2.html

ratings higher than the rest of the documents in the collection. In §2.4, we describe standard IR metrics for quantifying model performance given the ranked documents retrieved by the model and the corresponding assessor judgments for a given query.

> ✎ This tutorial focuses on text retrieval systems that rank either long documents or short answers in response to queries that are typically few terms in length.

Question-answering   Question-answering tasks may range from choosing between multiple choices (typically entities or binary true-or-false decisions) (Richardson *et al.*, 2013; Hermann *et al.*, 2015; Hill *et al.*, 2015; Weston *et al.*, 2015) to ranking spans of text or passages (Voorhees and Tice, 2000; Yang *et al.*, 2015; Rajpurkar *et al.*, 2016; Agichtein *et al.*, 2015; Ferrucci *et al.*, 2010), and may even include synthesizing textual responses by gathering evidence from one or more sources (Nguyen *et al.*, 2016b; Mitra *et al.*, 2016b). TREC question-answering experiments (Voorhees and Tice, 2000) has participating IR systems retrieve spans of text, rather than documents, in response to questions. IBM's DeepQA (Ferrucci *et al.*, 2010) system—behind the Watson project that famously demonstrated human-level performance on the American TV quiz show, "Jeopardy!"—also has a primary search phase, whose goal is to find as many potentially answer-bearing passages of text as possible. With respect to the question-answering task, the scope of this tutorial is limited to ranking answer containing passages in response to natural language questions or short query texts.

Retrieving short spans of text pose different challenges than ranking documents. Unlike the long body text of documents, single sentences or short passages tend to be on point with respect to a single topic. However, answers often tend to use different vocabulary than the one used to frame the question. For example, the span of text that contains the answer to the question "what year was Martin Luther King Jr.

born?" may not contain the term "year". However, the phrase "what year" implies that the correct answer text should contain a year—such as "1929" in this case. Therefore, IR systems that focus on the question-answering task need to model the patterns expected in the answer passage based on the intent of the question.

2.2 Desiderata of IR models

Before we describe any specific IR model, it is important for us to discuss some of the attributes that we desire from a good retrieval system. For any IR system, the relevance of the retrieved items to the input query is of foremost importance. But relevance measurements can be nuanced by the properties of robustness, sensitivity and efficiency that we expect the system to demonstrate. These attributes not only guide our model designs but also serve as yard sticks for comparing the different neural and non-neural approaches.

Semantic understanding   Most traditional approaches to ad-hoc retrieval count repetitions of the query terms in the document text. *Exact term matching* between query and document text, while simple, serves as a foundation for many IR systems. Different weighting and normalization schemes over these counts leads to a variety of TF-IDF models, such as BM25 (Robertson and Zaragoza, 2009). However, by only inspecting the query terms the IR model ignores all the evidence of *aboutness* from the rest of the document. So, when ranking for the query "Australia" only the occurrences of "Australia" in the document are considered—although the frequency of other terms like "Sydney" or "kangaroo" may be highly informative. In the case of the query "what channel are the seahawks on today", the query term "channel" provides hints to the IR model to pay attention to the occurrences of "ESPN" or "Sky Sports" in the document text—none of which appears in the query itself.

Semantic understanding, however, goes beyond mapping query terms to document terms (Li, Xu, *et al.*, 2014). A good IR model may consider the terms "hot" and "warm" related, as well as the terms "dog" and "puppy"—but must also distinguish that a user who submits the query

(a) Distribution of query impressions   **(b)** Distribution of document clicks
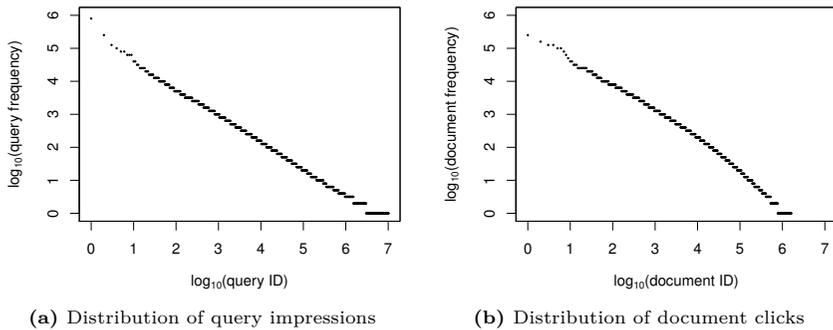
Figure 2.1: A Log-Log plot of frequency versus rank for query impressions and document clicks in the AOL query logs (Pass *et al.*, 2006). The plots highlight that these quantities follow a Zipfian distribution.

"hot dog" is not looking for a "warm puppy" (Levy, 2011). At the more ambitious end of the spectrum, semantic understanding would involve logical reasons by the IR system—so for the query "concerts during SIGIR" it associates a specific edition of the conference (the upcoming one) and considers both its location and dates when recommending concerts nearby during the correct week.

These examples motivate that IR models should have some latent representations of intent as expressed by the query and of the different topics in the document text—so that *inexact matching* can be performed that goes beyond lexical term counting.

Robustness to rare inputs   Query frequencies in most IR tasks follow a Zipfian distribution (Xie and O'Hallaron, 2002) (see Figure 2.1). In the publicly available AOL query logs (Pass *et al.*, 2006), for example, more than 70% of the distinct queries are seen only once in the period of three months from which the queries are sampled. In the same dataset, more than 50% of the distinct documents are clicked only once. A good IR method must be able to retrieve these infrequently searched-for documents and perform reasonably well on queries containing terms that appear extremely rarely, if ever, in its historical logs.

Many IR models that learn latent representations of text from data often naively assume a fixed size vocabulary. These models perform

poorly when the query consists of terms rarely (or never) seen during training. Even if the model does not assume a fixed vocabulary, the quality of the latent representations may depend heavily on how often the terms under consideration appear in the training dataset. *Exact matching* models, like BM25 (Robertson and Zaragoza, 2009), on the other hand can precisely retrieve documents containing rare terms.

Semantic understanding in an IR model cannot come at the cost of poor retrieval performance on queries containing rare terms. When dealing with a query such as "pekarovic land company" the IR model will benefit from considering exact matches of the rare term "pekarovic". In practice an IR model may need to effectively trade-off exact and inexact matching for a query term. However, the decision of when to perform exact matching can itself be informed by semantic understanding of the context in which the terms appear in addition to the terms themselves.

> ✏ Learning latent representation of text is important for dealing with vocabulary mismatch, but exact matching is also important to deal with rare terms and intents.

Robustness to corpus variance An interesting consideration for IR models is how well they perform on corpora whose distributions are different from the data that the model was trained on. Models like BM25 (Robertson and Zaragoza, 2009) have very few parameters and often demonstrate reasonable performance "out of the box" on new corpora with little or no additional tuning of parameters. Supervised deep learning models containing millions (or even billions) of parameters, on the other hand, are known to be more sensitive to distributional differences between training and evaluation data, and have been shown to be especially vulnerable to adversarial inputs (Szegedy *et al.*, 2013). The application of unsupervised term embeddings on collections and tasks that are different from the original data the representations were trained on is common in the literature. While these can be seen as

examples of successful transfer learning, there is also evidence (Diaz *et al.*, 2016) that term embeddings trained on collections distributionally closer to the test samples perform significantly better.

Some of the variances in performance of deep models on new corpora is offset by better retrieval on the test corpus that is distributionally closer to the training data, where the model may have picked up crucial corpus specific patterns. For example, it may be understandable if a model that learns term representations based on the text of Shakespeare's Hamlet is effective at retrieving passages relevant to a search query from The Bard's other works, but performs poorly when the retrieval task involves a corpus of song lyrics by Jay-Z. However, the poor performances on new corpus can also be indicative that the model is overfitting, or suffering from the Clever Hans[2] effect (Sturm, 2014). For example, an IR model trained on recent news corpus may learn to associate "Theresa May" with the query "uk prime minister" and as a consequence may perform poorly on older TREC datasets where the connection to "John Major" may be more appropriate.

ML models that are hyper-sensitive to corpus distributions may be vulnerable when faced with unexpected changes in distributions in the test data (Cohen *et al.*, 2018). This can be particularly problematic when the test distributions naturally evolve over time due to underlying changes in the user population or behaviour. The models may need to be re-trained periodically or designed to be invariant to such changes.

Robustness to variable length inputs   Typical text collections contain documents of varied lengths (see Figure 2.2). A good IR system must be able to deal with documents of different lengths without over-retrieving either long or short documents. Relevant documents may contain irrelevant sections, and the relevant content may either be localized, or spread over multiple sections in the document. Document length normalization is well-studied in the context of IR models (*e.g.*, pivoted length normalization (Singhal *et al.*, 1996)), and this existing research should inform the design of any new IR models.

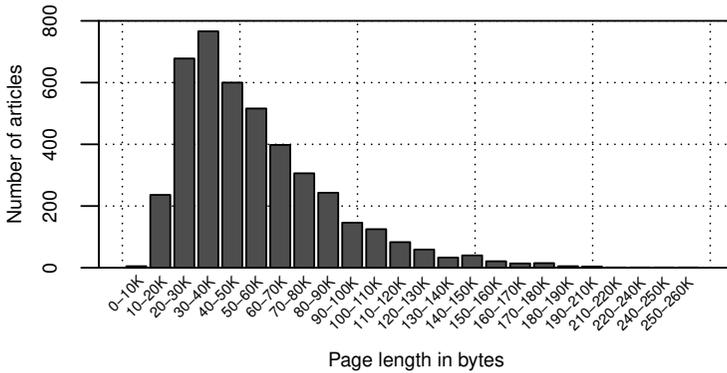[2]https://en.wikipedia.org/wiki/Clever_Hans

Figure 2.2: Distribution of document length (in bytes) of Wikipedia featured articles as of June 30, 2014. Source: https://en.wikipedia.org/wiki/Wikipedia:Featured_ articles/By_length.

Robustness to errors in input   No IR system should assume error-free inputs—neither when considering the user query nor when inspecting the documents in the collection. While traditional IR models have typically involved specific components for error correction—such as automatic spell corrections over queries—new IR models may adopt different strategies towards error handling by operating at the character-level and/or by learning better representations from noisy texts.

Sensitivity to context   Retrieval in the wild can leverage many implicit and explicit context information.[3] The query "weather" may refer to the weather in Seattle or in London depending on where the user is located. An IR model may retrieve different results for the query "decorations" depending on the current season. The query "giants match highlights" may be better disambiguated if the system knows whether the user is a fan of baseball or American football, whether she is located on the East or the West coast of USA, or if the model has knowledge of recent sport fixtures. In conversational IR systems, the correct response to the

---

[3]In *proactive retrieval* scenarios (Liebling *et al.*, 2012; Song and Guo, 2016; Benetka *et al.*, 2017; Shokouhi and Guo, 2015), the retrieval can even be triggered based solely on implicit context without any explicit query submission from the user.

question "When did she become the prime minister?" would depend on disambiguating the correct entity based on the context of references made in the previous turns of the conversation. Relevance in many applications is, therefore, situated in the user and task context, and is an important consideration in the design of IR systems.

Efficiency   Efficiency of retrieval is one of the salient points of any retrieval system. A typical commercial Web search engine may deal with tens of thousands of queries per second[4]—retrieving results for each query from an index containing billions of documents. Search engines typically involve large multi-tier architectures and the retrieval process generally consists of multiple stages of pruning the candidate set of documents (Matveeva *et al.*, 2006; Wang *et al.*, 2011). The IR model at the bottom of this *telescoping* setup may need to sift through billions of documents—while the model at the top may only need to re-rank between tens of promising documents. The retrieval approaches that are suitable at one level of the stack may be highly impractical at a different step—models at the bottom need to be *fast* but mostly focus on eliminating irrelevant or junk results, while models at the top tend to develop more sophisticated notions of *relevance*, and focus on distinguishing between documents that are much closer on the relevance scale. So far, much of the focus on neural IR approaches have been limited to re-ranking top-n documents.

✎   A telescoping setup involves a multi-tier architecture, where different IR models prune the set of candidate documents further at each stage. The model at the beginning needs to retrieve from the full collection, whereas the final model needs to re-rank only the top candidates.

---

[4]http://www.internetlivestats.com/one-second/#google-band

Table 2.1: Notation used in this tutorial.

Meaning	Notation
Single query	q
Single document	d
Set of queries	Q
Collection of documents	D
Term in query q	t_q
Term in document d	t_d
Full vocabulary of all terms	T
Set of ranked results retrieved for query q	R_q
Result tuple (document d at rank i)	$\langle i, d \rangle$, where $\langle i, d \rangle \in R_q$
Relevance label of document d for query q	$rel_q(d)$
d_i is more relevant than d_j for query q	$rel_q(d_i) > rel_q(d_j)$, or $d_i \succ_q d_j$
Frequency of term t in document d	$tf(t, d)$
Number of documents that contain term t	$df(t)$
Vector representation of text z	$\vec{v}_z$
Probability function for an event $\mathcal{E}$	$p(\mathcal{E})$

While this list of desired attributes of an IR model is in no way complete, it serves as a reference for comparing many of the neural and non-neural approaches described in the rest of this tutorial.

2.3 Notation

We adopt some common notation for this tutorial shown in Table 2.1. We use lower-case to denote vectors (*e.g.*, $\vec{x}$) and upper-case for tensors of higher dimensions (*e.g.*, X). The ground truth $rel_q(d)$ in Table 2.1 may be based on either manual relevance annotations or be implicitly derived from user behaviour on SERP (*e.g.*, from clicks).

2.4 Metrics

A large number of IR studies (Granka *et al.*, 2004; Joachims *et al.*, 2005; Guan and Cutrell, 2007; Joachims *et al.*, 2007; Diaz *et al.*, 2013; Mitra *et al.*, 2014; Hofmann *et al.*, 2014; Lagun *et al.*, 2014) have demonstrated

that users of retrieval systems tend to pay attention mostly to top-ranked results. IR metrics, therefore, focus on rank-based comparisons of the retrieved result set R to an ideal ranking of documents, as determined by manual judgments or implicit feedback from user behaviour data. These metrics are typically computed at a rank position, say k, and then averaged over all queries in the test set. Unless otherwise specified, R refers to the top-k results retrieved by the model. Next, we describe a few standard metrics used in IR evaluations.

> IR metrics focus on rank-based evaluation of retrieved results using ground truth information, as determined by manual judgments or implicit feedback from user behaviour data.

Precision and recall   Precision and recall both compute the fraction of relevant documents retrieved for a query q, but with respect to the total number of documents in the retrieved set R_q and the total number of relevant documents in the collection D, respectively. Both metrics assume that the relevance labels are binary.

$$Precision_q = \frac{\sum_{\langle i,d \rangle \in R_q} rel_q(d)}{|R_q|} \tag{2.1}$$

$$Recall_q = \frac{\sum_{\langle i,d \rangle \in R_q} rel_q(d)}{\sum_{d \in D} rel_q(d)} \tag{2.2}$$

Mean reciprocal rank (MRR)   Mean reciprocal rank (Craswell, 2009) is also computed over binary relevance judgments. It is given as the reciprocal rank of the first relevant document averaged over all queries.

$$RR_q = \max_{\langle i,d \rangle \in R_q} \frac{rel_q(d)}{i} \tag{2.3}$$

Mean average precision (MAP) The average precision (Zhu, 2004) for a ranked list of documents R is given by,

$$AveP_q = \frac{\sum_{\langle i,d \rangle \in R_q} Precision_{q,i} \times rel_q(d)}{\sum_{d \in D} rel_q(d)} \tag{2.4}$$

where, $Precision_{q,i}$ is the precision computed at rank i for the query q. The average precision metric is generally used when relevance judgments are binary, although variants using graded judgments have also been proposed (Robertson *et al.*, 2010). The mean of the average precision over all queries gives the MAP score for the whole set.

Normalized discounted cumulative gain ($NDCG$) There are a few different variants of the discounted cumulative gain (DCG_q) metric (Järvelin and Kekäläinen, 2002) which can be used when graded relevance judgments are available for a query q—say, on a five-point scale between zero to four. One incarnation of this metric is as follows.

$$DCG_q = \sum_{\langle i,d \rangle \in R_q} \frac{2^{rel_q(d)} - 1}{\log_2(i+1)} \tag{2.5}$$

The ideal DCG ($IDCG_q$) is computed the same way but by assuming an ideal rank order for the documents up to rank k. The normalized DCG ($NDCG_q$) is then given by,

$$NDCG_q = \frac{DCG_q}{IDCG_q} \tag{2.6}$$

2.5 Traditional IR models

In this section, we introduce a few of the traditional IR approaches. The decades of insights from these IR models not only inform the design of our new neural based approaches, but these models also serve as important baselines for comparison. They also highlight the various desiderata that we expect the neural IR models to incorporate.

BM25 There is a broad family of statistical functions in IR that consider the number of occurrences of each query term in the document—*i.e.*, term-frequency (TF)—and the corresponding inverse document

frequency (IDF) of the same terms in the full collection (as an indicator of the informativeness of the term). One theoretical basis for such formulations is the probabilistic model of IR that yielded the popular BM25 (Robertson and Zaragoza, 2009) ranking function.

$$BM25(q,d) = \sum_{t_q \in q} idf(t_q) \cdot \frac{tf(t_q,d) \cdot (k_1 + 1)}{tf(t_q,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{avgdl}\right)} \quad (2.7)$$

where, $avgdl$ is the average length of documents in the collection D, and k_1 and b are parameters that are usually tuned on a validation dataset. In practice, k_1 is sometimes set to some default value in the range $[1.2, 2.0]$ and b as 0.75. The $idf(t)$ is computed as,

$$idf(t) = \log \frac{|D| - df(t) + 0.5}{df(t) + 0.5} \quad (2.8)$$

BM25 aggregates the contributions from individual terms but ignores any phrasal or proximity signals between the occurrences of the different query terms in the document. A variant of BM25 (Zaragoza *et al.*, 2004; Robertson *et al.*, 2004) also considers documents as composed of several fields (such as, title, body, and anchor texts).

Language modelling (LM)   In the language modelling based approach (Ponte and Croft, 1998; Hiemstra, 2001; Zhai and Lafferty, 2001), documents are ranked by the posterior probability $p(d|q)$.

$$p(d|q) = \frac{p(q|d).p(d)}{\sum_{\bar{d} \in D} p(q|\bar{d}).p(\bar{d})} \quad (2.9)$$

$$\propto p(q|d).p(d) \quad (2.10)$$

$$= p(q|d), \text{ assuming p(d) is uniform} \quad (2.11)$$

$$= \prod_{t_q \in q} p(t_q|d) \quad (2.12)$$

$\hat{p}(\mathcal{E})$ is the maximum likelihood estimate (MLE) of the probability of event $\mathcal{E}$, and $p(q|d)$ indicates the probability of generating query q by randomly sampling terms from document d. In its simplest form, we can estimate $p(t_q|d)$ by,

$$p(t_q|d) = \frac{tf(t_q,d)}{|d|} \quad (2.13)$$

However, most formulations of language modelling based retrieval typically employ some form of smoothing (Zhai and Lafferty, 2001) by sampling terms from both the document d and the full collection D. The two common smoothing methods are:

1. Jelinek-Mercer smoothing (Jelinek and Mercer, 1980)

$$p(t_q|d) = \left(\lambda \frac{tf(t_q, d)}{|d|} + (1 - \lambda) \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right) \qquad (2.14)$$

2. Dirichlet Prior Smoothing (MacKay and Peto, 1995)

$$p(t_q|d) = \left(tf(t_q, d) + \mu \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right) / \left(|d| + \mu \right) \qquad (2.15)$$

Both TF-IDF and language modelling based approaches estimate document relevance based on the count of only the query terms in the document. The position of these occurrences and the relationship with other terms in the document are ignored.

Translation models   Berger and Lafferty (1999) proposed an alternative method to estimate $p(t_q|d)$ in the language modelling based IR approach (Equation (2.12)), by assuming that the query q is being generated via a "translation" process from the document d.

$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d) \cdot p(t_d|d) \qquad (2.16)$$

The $p(t_q|t_d)$ component allows the model to garner evidence of relevance from non-query terms in the document. Berger and Lafferty (1999) propose to estimate $p(t_q|t_d)$ from query-document paired data similar to techniques in statistical machine translation (Brown *et al.*, 1990; Brown *et al.*, 1993)—but other approaches for estimation have also been explored (Zuccon *et al.*, 2015).

Dependence model   None of the three IR models described so far consider proximity between query terms. To address this, Metzler and

Croft (2005) proposed a linear model over proximity-based features.

$$DM(q,d) = (1 - \lambda_{ow} - \lambda_{uw}) \sum_{t_q \in q} \log \left((1 - \alpha_d) \frac{tf(t_q, d)}{|d|} + \alpha_d \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right)$$

$$+ \lambda_{ow} \sum_{c_q \in ow(q)} \log \left((1 - \alpha_d) \frac{tf_{\#1}(c_q, d)}{|d|} + \alpha_d \frac{\sum_{\bar{d} \in D} tf_{\#1}(c_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right)$$

$$+ \lambda_{uw} \sum_{c_q \in uw(q)} \log \left((1 - \alpha_d) \frac{tf_{\#uwN}(c_q, d)}{|d|} + \alpha_d \frac{\sum_{\bar{d} \in D} tf_{\#uwN}(c_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right)$$

$$(2.17)$$

where, $ow(q)$ and $uw(q)$ are the set of all contiguous n-grams (or phrases) and the set of all bags of terms that can be generated from query q. $tf_{\#1}$ and $tf_{\#uwN}$ are the ordered-window and unordered-window operators from Indri (Strohman *et al.*, 2005). Finally, λ_{ow} and λ_{uw} are the tuneable parameters of the model.

Pseudo relevance feedback (PRF)   PRF-based methods—*e.g.*, Relevance Models (RM) (Lavrenko, 2008; Lavrenko and Croft, 2001)—typically demonstrate strong performance at the cost of executing an additional round of retrieval. The set of ranked documents R_1 from the first round of retrieval is used to select expansion terms to augment the query which is used to retrieve a new ranked set of documents R_2 that is presented to the user.

The underlying approach to scoring a document in RM is by computing the KL divergence (Lafferty and Zhai, 2001) between the query language model θ_q and the document language model θ_d.

$$score(q,d) = -\sum_{t \in T} p(t|\theta_q) \log \frac{p(t|\theta_q)}{p(t|\theta_d)} \qquad (2.18)$$

Without PRF,

$$p(t|\theta_q) = \frac{tf(t,q)}{|q|} \qquad (2.19)$$

But under the RM3 (Abdul-Jaleel *et al.*, 2004) formulation the new query language model $\bar{\theta}_q$ is estimated by,

$$p(t|\bar{\theta}_q) = \alpha \frac{tf(t,q)}{|q|} + (1-\alpha) \sum_{d \in R_1} p(t|\theta_d)p(d) \prod_{\bar{t} \in q} p(\bar{t}|\theta_d) \qquad (2.20)$$

Besides language models, PRF based query expansion has also been explored in the context of other retrieval approaches (*e.g.*, (Robertson *et al.*, 1996; Miao *et al.*, 2012)). By expanding the query using the results from the first round of retrieval PRF based approaches tend to be more robust to the vocabulary mismatch problem plaguing many other traditional IR methods.

✎ TF-IDF and language modelling based approaches only consider the count of query term occurrences in the document. Dependence model considers phrasal matches. Translation and PRF models can deal with vocabulary mismatch between query and document.

2.6   Neural approaches to IR

Document ranking comprises of performing three primary steps—generate a representation of the query that specifies the information need, generate a representation of the document that captures the distribution over the information contained, and match the query and the document representations to estimate their mutual relevance. All existing neural approaches to IR can be broadly categorized based on whether they influence the query representation, the document representation, or in estimating relevance. A neural approach may impact one or more of these stages shown in Figure 2.3.

Neural networks are useful as learning to rank models as we will discuss in §5. In these models, a joint representation of query and
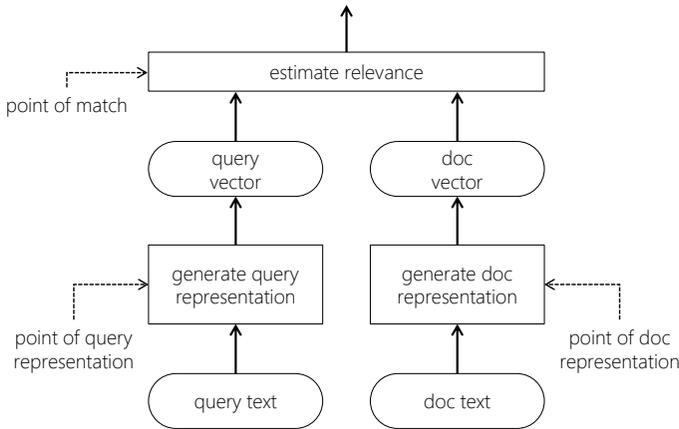
Figure 2.3: Document ranking typically involves a query and a document representation steps, followed by a matching stage. Neural models can be useful either for generating good representations or in estimating relevance, or both.

document is generated using manually designed features and the neural network is used only at the *point of match* to estimate relevance, as shown in Figure 2.4a. In §7.4, we will discuss DNN models, such as (Guo *et al.*, 2016a; Mitra *et al.*, 2017a), that estimate relevance based on patterns of exact query term matches in the document. Unlike traditional learning to rank models, however, these architectures (shown in Figure 2.4b) depend less on manual feature engineering and more on automatically detecting regularities in good matching patterns.

> ✎ Neural IR models can be categorized based on whether they influence the query representation, the document representation, the relevance estimation, or a combination of these steps.
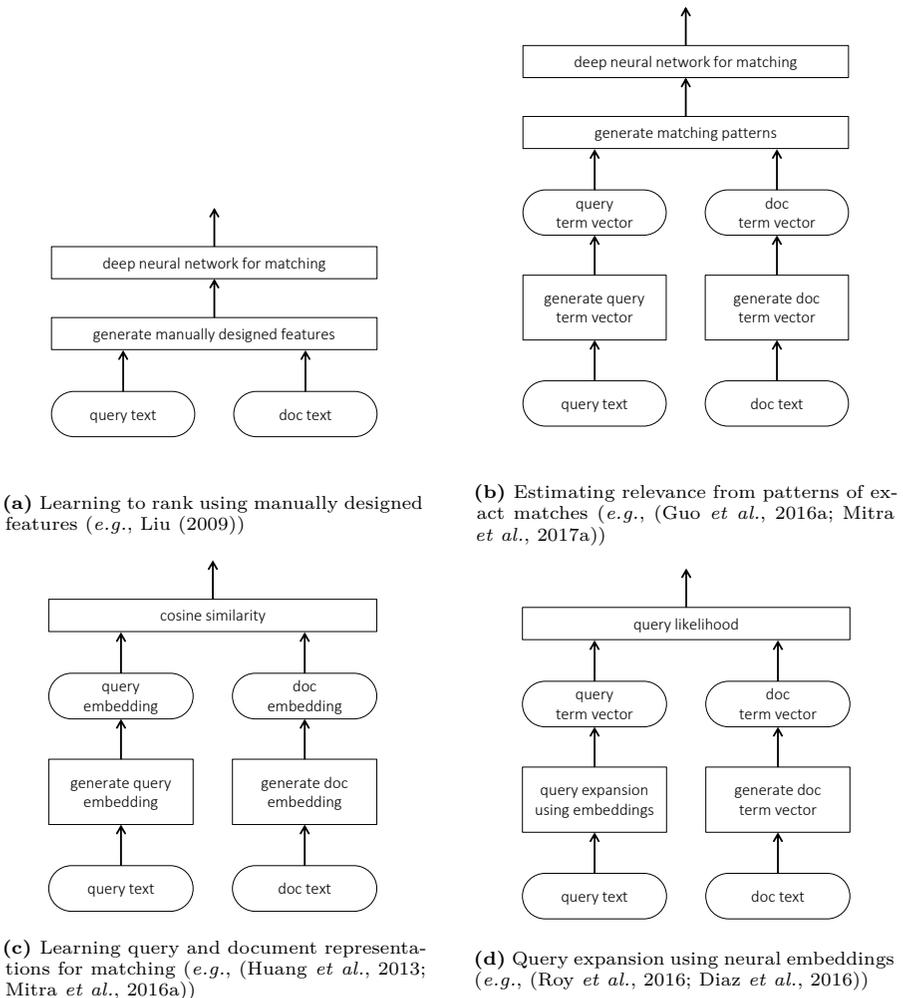
(a) Learning to rank using manually designed features (*e.g.*, Liu (2009))

(b) Estimating relevance from patterns of exact matches (*e.g.*, (Guo *et al.*, 2016a; Mitra *et al.*, 2017a))

(c) Learning query and document representations for matching (*e.g.*, (Huang *et al.*, 2013; Mitra *et al.*, 2016a))

(d) Query expansion using neural embeddings (*e.g.*, (Roy *et al.*, 2016; Diaz *et al.*, 2016))

Figure 2.4: Examples of different neural approaches to IR. In (a) and (b) the neural network is only used at the point of matching, whereas in (c) the focus is on learning effective representations of text using neural methods. Neural models can also be used to expand or augment the query before applying traditional IR techniques, as shown in (d).

In contrast, many (shallow and deep) neural IR models depend on learning useful low-dimensional vector representations—or *embeddings*— of query and document text, and using them within traditional IR

models or in conjunction with simple similarity metrics (*e.g.*, cosine similarity). These models shown in Figure 2.4c may learn the embeddings by optimizing directly for the IR task (*e.g.*, (Huang *et al.*, 2013)), or in an unsupervised setting (*e.g.*, (Mitra *et al.*, 2016a)). Finally, Figure 2.4d shows IR approaches where the neural models are used for query expansion (Diaz *et al.*, 2016; Roy *et al.*, 2016).

While the taxonomy of neural approaches described in this section is rather simple, it does provide an intuitive framework for comparing the different neural approaches in IR and highlights the similarities and distinctions between these different techniques.

3

Unsupervised learning of term representations

3.1 A tale of two representations

Vector representations are fundamental to both information retrieval and machine learning. In IR, terms are typically the smallest unit of representation for indexing and retrieval. Therefore, many IR models—both non-neural and neural—focus on learning good vector representations of terms. Different vector representations exhibit different levels of generalization—some consider every term as a distinct entity while others learn to identify common attributes. Different representation schemes derive different notions of similarity between terms from the definition of the corresponding vector spaces. Some representations operate over fixed-size vocabularies, while the design of others obviate such constraints. They also differ on the properties of compositionality that defines how representations for larger units of information, such as passages and documents, can be derived from individual term vectors. These are some of the important considerations for choosing a term representation suitable for a specific task.

Local representations   Under local (or *one-hot*) representations, every term in a fixed size vocabulary T is represented by a binary vector
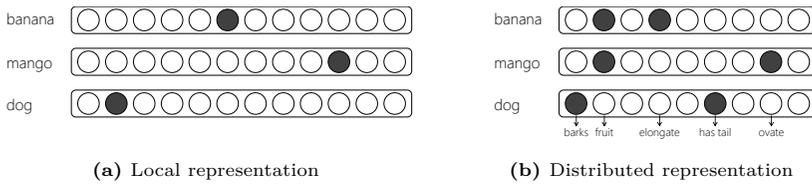
(a) Local representation   **(b)** Distributed representation

Figure 3.1: Under local representations the terms "banana", "mango", and "dog" are distinct items. But distributed vector representations may recognize that "banana" and "mango" are both fruits, but "dog" is different.

$\vec{v} \in \{0,1\}^{|T|}$, where only one of the values in the vector is one and all the others are set to zero. Each position in the vector $\vec{v}$ corresponds to a term. The term "banana", under this representation, is given by a vector that has the value one in the position corresponding to "banana" and zero everywhere else. Similarly, the terms "mango" and "dog" are represented by setting different positions in the vector to one. Figure 3.1a highlights that under this scheme each term is a unique entity, and "banana" is as distinct from "dog" as it is from "mango". Terms outside of the vocabulary either have no representation or are denoted by a special "UNK" symbol under this scheme.

Distributed representations   Under distributed representations every term is represented by a vector $\vec{v} \in \mathbb{R}^{|k|}$. $\vec{v}$ can be a sparse or a dense vector—a vector of hand-crafted features or a latent representation in which the individual dimensions are not interpretable in isolation. The key underlying hypothesis for any distributed representation scheme, however, is that by representing a term by its attributes allows for defining some notion of similarity between the different terms based on the chosen properties. For example, in Figure 3.1b "banana" is more similar to "mango" than "dog" because they are both fruits, but yet different because of other properties that are not shared between the two, such as shape.

banana

(a) In-document features

banana

(b) Neighbouring-term features

banana

(c) Neighbouring-term w/ distance features
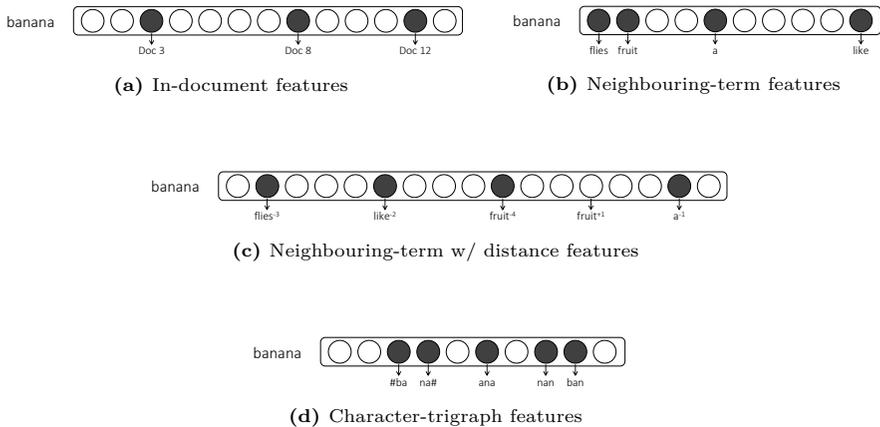
banana

(d) Character-trigraph features

Figure 3.2: Examples of different feature-based distributed representations of the term "banana". The representations in (a), (b), and (c) are based on external contexts in which the term frequently occurs, while (d) is based on properties intrinsic to the term. The representation scheme in (a) depends on the documents containing the term while the scheme shown in (b) and (c) depends on other terms that appears in its neighbourhood. The scheme (b) ignores inter-term distances. Therefore, in the sentence "Time flies like an arrow; fruit flies like a banana", the feature "fruit" describes both the terms "banana" and "arrow". However, in the representation scheme of (c) the feature "fruit^{-4}" is positive for "banana", and the feature "fruit^{+1}" for "arrow".

> ✎ Under a local or one-hot representation every item is distinct. But when items have distributed or feature based representation, then the similarity between two items is determined based on the similarity between their features.

A key consideration in any feature based distributed representation is the choice of the features themselves. One approach involves representing terms by features that capture their distributional properties. This is motivated by the *distributional hypothesis* (Harris, 1954) that states that

terms that are used (or occur) in similar context tend to be semantically similar. Firth (1957) famously purported this idea of *distributional semantics*[1] by stating "*a word is characterized by the company it keeps*". However, the distribution of different types of context may model different semantics of a term. Figure 3.2 shows three different sparse vector representations of the term "banana" corresponding to different distributional feature spaces—documents containing the term (*e.g.*, LSA (Deerwester *et al.*, 1990)), neighbouring terms in a window (*e.g.*, HAL (Lund and Burgess, 1996), COALS (Rohde *et al.*, 2006), and (Bullinaria and Levy, 2007)), and neighbouring terms with distance (*e.g.*, (Levy *et al.*, 2014)). Finally, Figure 3.2d shows a vector representation of "banana" based on the character trigraphs in the term itself—instead of external contexts in which the term occurs. In §3.2 we will discuss how choosing different distributional features for term representation leads to different nuanced notions of semantic similarity between them. When the vectors are high-dimensional, sparse, and based on observable features we refer to them as *observed* (or *explicit*) vector representations (Levy *et al.*, 2014). When the vectors are dense, small ($k \ll |T|$), and learnt from data then we instead refer to them as *latent* vector spaces, or *embeddings*. In both observed and latent vector spaces, several distance metrics can be used to define the similarity between terms, although cosine similarity is commonly used.

$$sim(\vec{v}_i, \vec{v}_j) = cos(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i^{\mathsf{T}} \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|} \tag{3.1}$$

Most embeddings are learnt from observed features, and hence the discussions in §3.2 about different notions of similarity are also relevant to the embedding models. In §3.3 and §3.4 we discuss observed and latent space representations. In the context of neural models, distributed representations generally refer to learnt embeddings. The idea of 'local' and 'distributed' representations has a specific significance in the context of neural networks. Each concept, entity, or term can be represented

---

[1]Readers should take note that while many distributed representations take advantage of *distributional* properties, the two concepts are not synonymous. A term can have a distributed representation based on non-distributional features—*e.g.*, parts of speech classification and character trigraphs in the term.
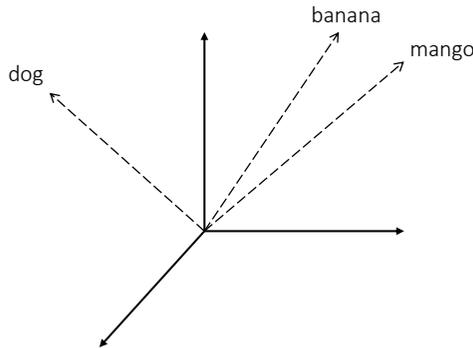
Figure 3.3: A vector space representation of terms puts "banana" closer to "mango" because they share more common attributes than "banana" and "dog".

within a neural network by the activation of a single neuron (local representation) or by the combined pattern of activations of several neurons (distributed representation) (Hinton, 1984).

Finally, with respect to *compositionality*, it is important to understand that distributed representations of items are often derived from local or distributed representation of its parts. For example, a document can be represented by the sum of the one-hot vectors or embeddings corresponding to the terms in the document. The resultant vector, in both cases, corresponds to a distributed bag-of-terms representation. Similarly, the character trigraph representation of terms in Figure 3.2d is simply an aggregation over the one-hot representations of the constituent trigraphs.

3.2   Notions of similarity

Any vector representation inherently defines some notion of relatedness between terms. Is "Seattle" closer to "Sydney" or to "Seahawks"? The answer depends on the type of relationship we are interested in. If we want terms of similar *type* to be closer, then "Sydney" is more similar to "Seattle" because they are both cities. However, if we are interested to find terms that co-occur in the same document or passage, then "Seahawks"—Seattle's football team—should be closer. The former

Table 3.1: A toy corpus of short documents that we consider for the discussion on different notions of similarity between terms under different distributed representations. The choice of the feature space that is used for generating the distributed representation determines which terms are closer in the vector space, as shown in Figure 3.4.

Sample documents			
doc 01	Seattle map	doc 09	Denver map
doc 02	Seattle weather	doc 10	Denver weather
doc 03	Seahawks jerseys	doc 11	Broncos jerseys
doc 04	Seahawks highlights	doc 12	Broncos highlights
doc 05	Seattle Seahawks Wilson	doc 13	Denver Broncos Lynch
doc 06	Seattle Seahawks Sherman	doc 14	Denver Broncos Sanchez
doc 07	Seattle Seahawks Browner	doc 15	Denver Broncos Miller
doc 08	Seattle Seahawks Ifedi	doc 16	Denver Broncos Marshall

represents a *typical*, or type-based notion of similarity while the latter exhibits a more *topical* sense of relatedness.

If we want to compare "Seattle" with "Sydney" and "Seahawks based on their respective vector representations, then the underlying feature space needs to align with the notion of similarity that we are interested in. It is, therefore, important for the readers to build an intuition about the choice of features and the notion of similarity they encompass. This can be demonstrated by using a toy corpus, such as the one in Table 3.1. Figure 3.4a shows that the "in documents" features naturally lend to a topical sense of similarity between the terms, while the "neighbouring terms with distances" features in Figure 3.4c gives rise to a more typical notion of relatedness. Using "neighbouring terms" without the inter-term distances as features, however, produces a mixture of topical and typical relationships. This is because when the term distances (denoted as superscripts) are considered in the feature definition then the document "Seattle Seahawks Wilson" produces the bag-of-features $\{Seahawks^{+1}, Wilson^{+2}\}$ for "Seattle" which is non-overlapping with the bag-of-features $\{Seattle^{-1}, Wilson^{+1}\}$ for "Seahawks". However, when the feature definition ignores the term-distances then there is a partial overlap between the bag-of-features $\{Seahawks, Wilson\}$ and $\{Seattle, Wilson\}$ corresponding to "Seattle"

and "Seahawks", respectively. The overlap increases when a larger window-size over the neighbouring terms is employed pushing the notion of similarity closer to a topical definition. This effect of the windows size on the latent vector space was reported by Levy and Goldberg (2014) in the context of term embeddings.

✎ Different vector representations capture different notions of similarity between terms. "Seattle" may be closer to either "Sydney" (*typically* similar) or "Seahawks" (*topically* similar) depending on the choice of vector dimensions.

Readers should note that the set of all inter-term relationships goes beyond the two notions of typical and topical that we discuss in this section. For example, vector representations could cluster terms closer based on linguistic styles—*e.g.*, terms that appear in thriller novels versus in children's rhymes, or in British versus American English. However, the notions of typical and topical similarities frequently come up in discussions in the context of many IR and NLP tasks—sometimes under different names such as *Paradigmatic* and *Syntagmatic* relations[2]— and the idea itself goes back at least as far as Saussure (De Saussure, 1916; Harris, 2001; Chandler, 1994; Sahlgren, 2006).

3.3   Observed feature spaces

Observed feature space representations can be broadly categorized based on their choice of distributional features (*e.g.*, in documents,

---

[2]Interestingly, the notion of Paradigmatic (typical) and Syntagmatic (topical) relationships show up almost universally—not just in text. In vision, for example, the different images of "nose" are typically similar to each other, while sharing topical relationship with images of "eyes" and "ears". Curiously, Barthes (1977) extended the analogy to garments. Paradigmatic relationships exist between items of the same type (*e.g.*, different style of boots) and the proper Syntagmatic juxtaposition of items from these different Paradigms—from hats to boots—forms a fashionable ensemble.
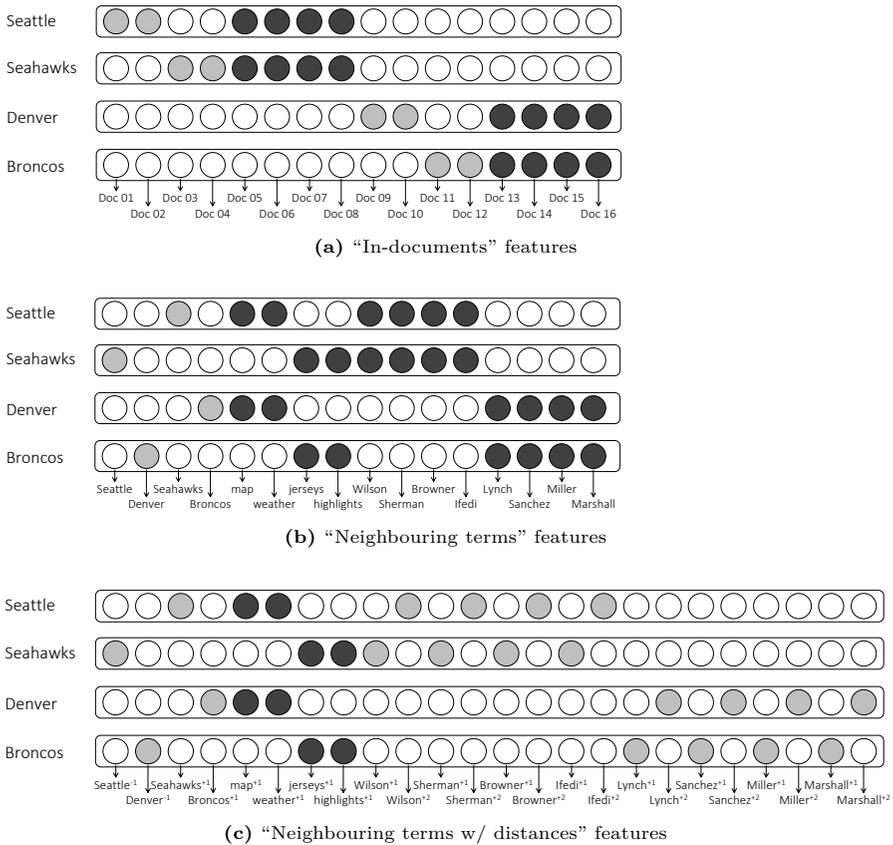
(a) "In-documents" features

(b) "Neighbouring terms" features

(c) "Neighbouring terms w/ distances" features

Figure 3.4: The figure shows different distributed representations for the four terms—"Seattle", "Seahawks", "Denver", and "Broncos"—based on the toy corpus in Table 3.1. Shaded circles indicate non-zero values in the vectors—the darker shade highlights the vector dimensions where more than one vector has a non-zero value. When the representation is based on the documents that the terms occur in then "Seattle" is more similar to "Seahawks" than to "Denver". The representation scheme in (a) is, therefore, more aligned with a topical notion of similarity. In contrast, in (c) each term is represented by a vector of neighbouring terms—where the distances between the terms are taken into consideration—which puts "Seattle" closer to "Denver" demonstrating a typical, or type-based, similarity. When the inter-term distances are ignored, as in (b), a mix of typical and topical similarities is observed. Finally, it is worth noting that neighbouring-terms based vector representations leads to similarities between terms that do not necessarily occur in the same document, and hence the term-term relationships are less sparse than when only in-document features are considered.
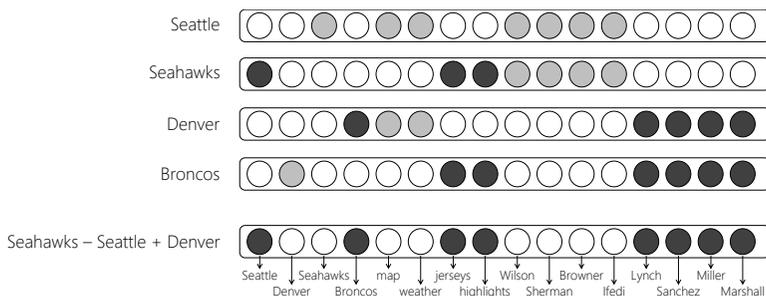
Figure 3.5: A visual demonstration of term analogies via simple vector algebra. The shaded circles denote non-zero values. Darker shade is used to highlight the non-zero values along the vector dimensions for which the output of $\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver}$ is positive. The output vector is closest to $\vec{v}_{Broncos}$ as shown in this toy example.

neighbouring terms with or without distances, *etc.*) and different weighting schemes (*e.g.*, TF-IDF, positive pointwise mutual information, *etc.*) applied over the raw counts. We direct the readers to (Turney and Pantel, 2010; Baroni and Lenci, 2010) which are good surveys of many existing observed vector representation schemes.

Levy *et al.* (2014) demonstrated that explicit vector representations are amenable to the term analogy task using simple vector operations. A term analogy task involves answering questions of the form "*man* is to *woman* as *king* is to _____?"—the correct answer to which in this case happens to be "queen". In NLP, term analogies are typically performed by simple vector operations of the following form followed by a nearest-neighbour search,

$$\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver} \approx \vec{v}_{Broncos} \qquad (3.2)$$

It may be surprising to some readers that the vector obtained by the simple algebraic operations $\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver}$ produces a vector close to the vector $\vec{v}_{Broncos}$. We present a visual intuition of why this works in practice in Figure 3.5, but we refer the readers to (Levy *et al.*, 2014; Arora *et al.*, 2015) for a more rigorous mathematical handling of this subject.

3.4   Latent feature spaces

While observed vector spaces based on distributional features can capture interesting relationships between terms, they have one big drawback—the resultant representations are highly sparse and high-dimensional. The number of dimensions, for example, may be the same as the vocabulary size, which is unwieldy for most practical tasks. An alternative is to learn lower dimensional representations that retains useful attributes from the observed feature spaces.

An *embedding* is a representation of items in a new space such that the properties of—and the relationships between—the items are preserved. Goodfellow *et al.* (2016) articulate that the goal of an embedding is to generate a *simpler* representation—where simplification may mean a reduction in the number of dimensions, an increase in the sparseness of the representation, disentangling the principle components of the vector space, or a combination of these goals. In the context of term embeddings, the explicit feature vectors—like those discussed in §3.3—constitutes the original representation. An embedding trained from these features assimilate the properties of the terms and the inter-term relationships observable in the original feature space.

> ✎ An embedding is a representation of items in a new space such that the properties of—and the relationships between—the items are preserved from the original representation.

Common approaches for learning embeddings include either factorizing the term-feature matrix (*e.g.*LSA (Deerwester *et al.*, 1990)) or using gradient descent based methods that try to predict the features given the term (*e.g.*, (Bengio *et al.*, 2003; Mikolov *et al.*, 2013a)). Baroni *et al.* (2014) empirically demonstrate that these feature-predicting models that learn lower dimensional representations, in fact, also perform better than explicit counting based models on different tasks—possibly due to

better generalization across terms—although some counter evidence the claim of better performances from embedding models have also been reported in the literature (Levy *et al.*, 2015).

The sparse feature spaces of §3.3 are easier to visualize and leads to more intuitive explanations—while their latent counterparts may be more practically useful. Therefore, it may be useful to *think sparse, but act dense* in many scenarios. In the rest of this section, we will describe some of these neural and non-neural latent space models.

Latent Semantic Analysis (LSA) LSA (Deerwester *et al.*, 1990) involves performing *singular value decomposition* (SVD) (Golub and Reinsch, 1970) on a term-document (or term-passage) matrix X to obtain its low-rank approximation (Markovsky, 2011). SVD on X involves solving $X = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix.[3]

$$
(\vec{t_i}^{\intercal}) \rightarrow
\begin{bmatrix}
x_{1,1} & \cdots & x_{1,|D|} \\
\vdots & \ddots & \vdots \\
x_{|T|,1} & \cdots & x_{|T|,|D|}
\end{bmatrix}
= (\vec{t_i}^{\intercal}) \rightarrow
\begin{bmatrix} \begin{bmatrix} \\ \vec{u}_1 \\ \\ \end{bmatrix} \cdots \begin{bmatrix} \\ \vec{u}_l \\ \\ \end{bmatrix} \end{bmatrix}
\cdot
\begin{bmatrix}
\sigma_1 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
0 & \cdots & \sigma_l
\end{bmatrix}
\cdot
\begin{bmatrix} [\ \vec{v}_1\ ] \\ \vdots \\ [\ \vec{v}_l\ ] \end{bmatrix}
$$

$$
\tag{3.3}
$$

where X is labeled $(\vec{d}_j)$, U, Σ, and $V^{\intercal}$ is labeled $(\vec{d}_j)$.

$\sigma_1, \ldots, \sigma_l$, $\vec{u}_1, \ldots, \vec{u}_l$, and $\vec{v}_1, \ldots, \vec{v}_l$ are the singular values, and the left and the right singular vectors, respectively. The k largest singular values—and corresponding singular vectors from U and V—is the rank k approximation of X ($X_k = U_k \Sigma_k V_k^T$) and $\Sigma_k \vec{t}_i$ is the embedding for the ith term.

While LSA operate on a term-document matrix, matrix factorization based approaches can also be applied to term-term matrices (Rohde *et al.*, 2006; Bullinaria and Levy, 2012; Lebret and Collobert, 2013).

Probabilistic Latent Semantic Analysis (PLSA) PLSA (Hofmann, 1999) learns low-dimensional representations of terms and documents

---

[3]The matrix visualization is taken from https://en.wikipedia.org/wiki/Latent_ semantic_analysis.

by modelling their co-occurrence $p(t, d)$ as follows,

$$p(t, d) = p(d) \sum_{c \in C} p(c|d) P(t|c) \tag{3.4}$$

where, C is the set of latent topics—and the number of topics $|C|$ is a hyperparameter of the model. Both $p(c|d)$ and $P(t|c)$ are modelled as multinomial distributions and their parameters are typically learned using the EM algorithm (Dempster *et al.*, 1977). After learning the parameters of the model, a term t_i can be represented as a distribution over the latent topics $[p(c_0|t_i), \ldots, p(c_{|C|-1}|t_i)]$. In a related approach called Latent Dirichlet Allocation (LDA) (Blei *et al.*, 2003), each document is represented by a Dirichlet prior instead of a fixed variable.

Neural term embedding models are typically trained by setting up a prediction task. Instead of factorizing the term-feature matrix—as in LSA—neural models are trained to predict the term from its features. The model learns dense low-dimensional representations in the process of minimizing the prediction error. These approaches are based on the *information bottleneck method* (Tishby *et al.*, 2000)—discussed more in §6.2—with the low-dimensional representations acting as the bottleneck. The training data may contain many instances of the same term-feature pair proportional to their frequency in the corpus (*e.g.*, word2vec (Mikolov *et al.*, 2013a)), or their counts can be pre-aggregated (*e.g.*, GloVe (Pennington *et al.*, 2014)).

✎ Instead of factorizing the term-feature matrix, neural models learn embeddings by setting up a feature prediction task and employ architectures motivated by the *information bottleneck* principle.

Word2vec   For word2vec (Mikolov *et al.*, 2013a; Mikolov *et al.*, 2013b; Mikolov *et al.*, 2013c; Goldberg and Levy, 2014; Rong, 2014), the features for a term are made up of its neighbours within a fixed size window over the text. The *skip-gram* architecture (see Figure 3.6a) is a

(a) Skip-gram

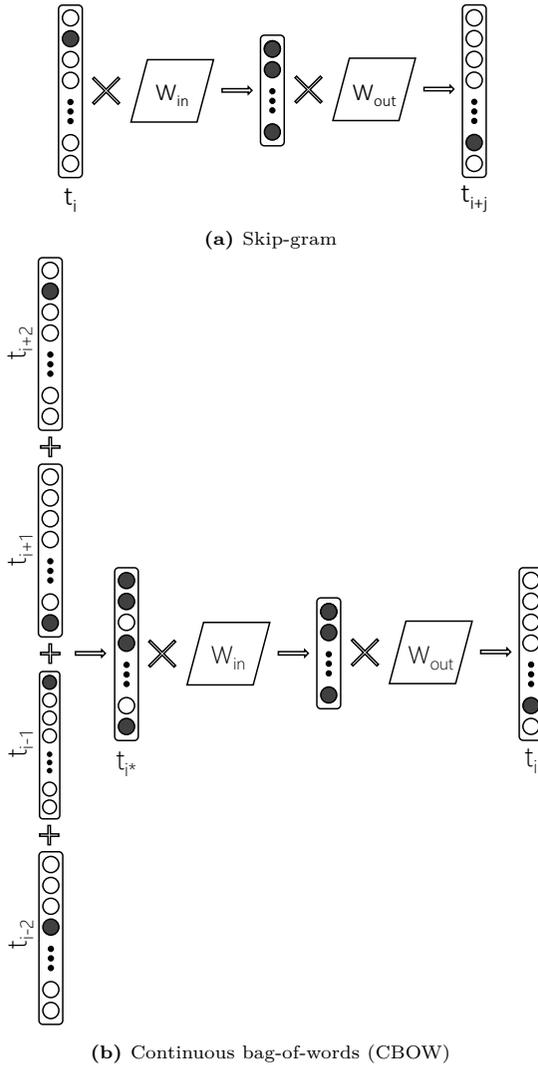(b) Continuous bag-of-words (CBOW)

Figure 3.6: The (a) skip-gram and the (b) continuous bag-of-words (CBOW) architectures of word2vec. The architecture is a neural network with a single hidden layer whose size is much smaller than that of the input and the output layers. Both models use one-hot representations of terms in the input and the output. The learnable parameters of the model comprise of the two weight matrices W_{in} and W_{out} that corresponds to the embeddings the model learns for the input and the output terms, respectively. The skip-gram model trains by minimizing the error in predicting a term given one of its neighbours. The CBOW model, in contrast, predicts a term from a bag of its neighbouring terms.

simple one hidden layer neural network. Both the input and the output of the model are one-hot vectors and the loss function is as follows,

$$\mathcal{L}_{skip-gram} = -\frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{-c \leq j \leq +c, j \neq 0} \log(p(t_{i+j}|t_i)) \qquad (3.5)$$

$$\text{where,} \qquad p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal (W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|} \exp\left((W_{out}\vec{v}_{t_k})^\intercal (W_{in}\vec{v}_{t_i})\right)} \qquad (3.6)$$

S is the set of all windows over the training text and c is the number of neighbours we want to predict on either side of the term t_i. The denominator for the softmax function for computing $p(t_{i+j}|t_i)$ sums over all the terms in the vocabulary. This is prohibitively costly and in practice either hierarchical-softmax (Morin and Bengio, 2005) or negative sampling is employed, which we discuss more in §5.2. Note that the model has two different weight matrices W_{in} and W_{out} that constitute the learnable parameters of the models. W_{in} gives us the IN embeddings corresponding to the input terms and W_{out} corresponds to the OUT embeddings for the output terms. Generally, only W_{in} is used and W_{out} is discarded after training. We discuss an IR application that makes use of both the IN and the OUT embeddings in §4.1.

The *continuous bag-of-words* (CBOW) architecture (see Figure 3.6b) is similar to the skip-gram model, except that the task is to predict the middle term given all the neighbouring terms in the window. The CBOW model creates a single training sample with the sum of the one-hot vectors of the neighbouring terms as input and the one-hot vector $\vec{v}_{t_i}$—corresponding to the middle term—as the expected output. Contrast this with the skip-gram model that creates $2 \times c$ samples by individually pairing each neighbouring term with the middle term. During training, the skip-gram model trains slower than the CBOW model (Mikolov *et al.*, 2013a) because it creates more training samples from the same windows of text.

$$\mathcal{L}_{CBOW} = -\frac{1}{|S|} \sum_{i=1}^{|S|} \log(p(t_i|t_{i-c}, \ldots, t_{i-1}, t_{i+1}, \ldots, t_{i+c})) \qquad (3.7)$$

Word2vec gained particular popularity for its ability to perform term analogies using simple vector algebra, similar to what we discussed
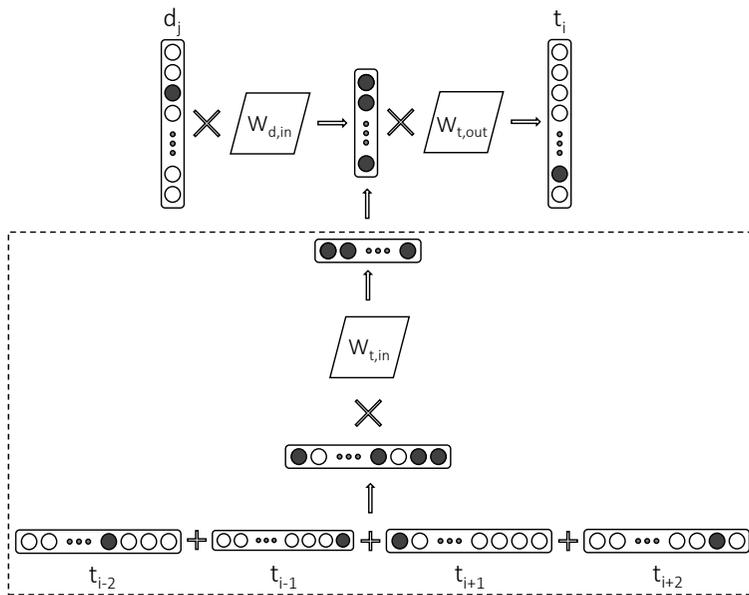
Figure 3.7: The paragraph2vec architecture as proposed by Le and Mikolov (2014) trains by predicting a term given a document (or passage) ID containing the term. By trying to minimize the prediction error, the model learns an embedding for the term as well as for the document. In some variants of the architecture, optionally the neighbouring terms are also provided as input—as shown in the dotted box.

in §3.3. For domains where the interpretability of the embeddings is important, Sun *et al.* (2016b) introduced an additional constraint in the loss function to encourage more sparseness in the learnt representations.

$$\mathcal{L}_{sparse-CBOW} = \mathcal{L}_{sparse-CBOW} - \lambda \sum_{t \in T} \|\vec{v}_t\|_1 \tag{3.8}$$

GloVe   The skip-gram model trains on individual term-neighbour pairs. If we aggregate all the training samples such that x_{ij} is the frequency of the pair $\langle t_i, t_j \rangle$ in the training data, then the loss function changes to,

$$\mathcal{L}_{skip-gram} = -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} x_{ij} \log(p(t_j|t_i)) \tag{3.9}$$

$$= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \frac{x_{ij}}{x_i} \log(p(t_j|t_i)) \tag{3.10}$$

$$= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \bar{p}(t_j|t_i) \log(p(t_j|t_i)) \tag{3.11}$$

$$= \sum_{i=1}^{|T|} x_i H(\bar{p}(t_j|t_i), p(t_j|t_i)) \tag{3.12}$$

$H(\ldots)$ is the cross-entropy error between the actual co-occurrence probability $\bar{p}(t_j|t_i)$ and the one predicted by the model $p(t_j|t_i)$. This is similar to the loss function for GloVe (Pennington *et al.*, 2014) if we replace the cross-entropy error with a squared-error and apply a saturation function $f(\ldots)$ over the actual co-occurrence frequencies.

$$\mathcal{L}_{GloVe} = -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} f(x_{ij})(log(x_{ij} - \vec{v}_{w_i}^{\intercal}\vec{v}_{w_j}))^2 \tag{3.13}$$

$$\tag{3.14}$$

where,

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & \text{if} x \leq x_{max} \\ 1, & \text{otherwise} \end{cases} \tag{3.15}$$

GloVe is trained using AdaGrad (Duchi *et al.*, 2011). Similar to word2vec, GloVe also generates two different (IN and OUT) embeddings, but unlike word2vec it generally uses the sum of the IN and the OUT vectors as the embedding for each term in the vocabulary.

Paragraph2vec   Following the popularity of word2vec (Mikolov *et al.*, 2013a; Mikolov *et al.*, 2013b), similar neural architectures (Le and Mikolov, 2014; Grbovic *et al.*, 2015b; Grbovic *et al.*, 2015a; Sun *et al.*, 2015; Ai *et al.*, 2016b; Ai *et al.*, 2016a) have been proposed that

trains on term-document co-occurrences. The training typically involves predicting a term given the ID of a document or a passage that contains the term. In some variants, as shown in Figure 3.7, neighbouring terms are also provided as input. The key motivation for training on term-document pairs is to learn an embedding that is more aligned with a topical notion of term-term similarity—which is often more appropriate for IR tasks. The term-document relationship, however, tends to be more sparse (Yan *et al.*, 2013)—including neighbouring term features may compensate for some of that sparsity. In the context of IR tasks, Ai *et al.* (2016b) and Ai *et al.* (2016a) proposed a number of IR-motivated changes to the original Paragraph2vec (Le and Mikolov, 2014) model training—including, document frequency based negative sampling and document length based regularization.

4

Term embeddings for IR

Traditional IR models use local representations of terms for query-document matching. The most straight-forward use case for term embeddings in IR is to enable *inexact* matching in the embedding space. In §2.2, we argued the importance of inspecting non-query terms in the document for garnering evidence of relevance. For example, even from a shallow manual inspection, it is possible to conclude that the passage in Figure 4.1a is *about* Albuquerque because it contains "metropolitan", "population", and "area" among other informative terms. On the other hand, the passage in Figure 4.1b contains "simulator", "interpreter", and "Altair" which suggest that the passage is instead more likely related to computers and technology. In traditional term counting based IR approaches these signals are often ignored.

Albuquerque is the most populous **city** in the U.S. state of **New Mexico**. The high-**altitude city** serves as the county seat of **Bernalillo** County, and it is situated in the **central** part of the state, straddling the **Rio Grande**. The **city population** is 557,169 as of the July 1, 2014 **population** estimate from the United States Census Bureau, and ranks as the 32nd-largest **city** in the U.S. The Albuquerque **metropolitan** statistical **area** (or MSA) has a **population** of 907,301 according to the United States Census Bureau's most recently available estimate for 2015.

(a) About Albuquerque

Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in **Albuquerque**, **New Mexico** in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

(b) Not about Albuquerque

Figure 4.1: Two passages both containing exactly a single occurrence of the query term "Albuquerque". However, the passage in (a) contains other terms such as "population" and "area" that are relevant to a description of the city. In contrast, the terms in passage (b) suggest that it is unlikely to be about the city, and only mentions the city potentially in a different context.

> ✎ Term embeddings can be useful for inexact matching—either by deriving latent vector representations of the query and the document text for matching, or as a mechanism for selecting additional terms for query expansion.

Unsupervised term embeddings can be incorporated into existing IR approaches for inexact matching. These approaches can be broadly categorized as those that compare the query with the document directly in the embedding space; and those that use embeddings to generate

suitable query expansion candidates from a global vocabulary and then perform retrieval based on the expanded query. We discuss both these classes of approaches in the remainder of this section.

4.1   Query-document matching

One strategy for using term embeddings in IR involves deriving a dense vector representation for the query and the document from the embeddings of the individual terms in the corresponding texts. The term embeddings can be aggregated in different ways, although using the *average word (or term) embeddings* (AWE) is quite common (Kiros *et al.*, 2014; Le and Mikolov, 2014; Vulić and Moens, 2015; Mitra *et al.*, 2016a; Nalisnick *et al.*, 2016; Kenter *et al.*, 2016; Sun *et al.*, 2016a). Non-linear combinations of term vectors—such as using Fisher Kernel Framework (Clinchant and Perronnin, 2013)—have also been explored, as well as other families of aggregate functions of which AWE has been shown to be a special case (Zamani and Croft, 2016b).

The query and the document embeddings themselves can be compared using a variety of similarity metrics, such as cosine similarity or dot-product. For example,

$$sim(q, d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^\intercal \vec{v}_d}{\|\vec{v}_q\| \|\vec{v}_d\|} \tag{4.1}$$

$$\text{where,} \quad \vec{v}_q = \frac{1}{|q|} \sum_{t_q \in q} \frac{\vec{v}_{t_q}}{\|\vec{v}_{t_q}\|} \tag{4.2}$$

$$\vec{v}_d = \frac{1}{|d|} \sum_{t_d \in d} \frac{\vec{v}_{t_d}}{\|\vec{v}_{t_d}\|} \tag{4.3}$$

An important consideration here is the choice of the term embeddings that is appropriate for the retrieval scenario. While, LSA (Deerwester *et al.*, 1990), word2vec (Mikolov *et al.*, 2013b), and GloVe (Pennington *et al.*, 2014) are commonly used—it is important to understand how the notion of inter-term similarity modelled by a specific vector space may influence its performance on a retrieval task. In the example in Figure 4.1, we want to rank documents that contains related terms—such as "population" or "area"—higher. These terms are topically

Table 4.1: Different nearest neighbours in the word2vec embedding space based on whether we compute IN-IN, OUT-OUT, or IN-OUT similarities between the terms. The examples are from (Nalisnick *et al.*, 2016; Mitra *et al.*, 2016a) where the word2vec embeddings are trained on search queries. Training on short query text, however, makes the inter-term similarity more pronouncedly typical (where, "Yale" is closer to "Harvard" and "NYU") when both terms are represented using their IN vectors. In contrast, the IN-OUT similarity (where, "Yale" is closer to "faculty" and "alumni") mirrors more the topical notions of relatedness.

yale			seahawks		
IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT
yale	yale	yale	seahawks	seahawks	seahawks
harvard	uconn	faculty	49ers	broncos	highlights
nyu	harvard	alumni	broncos	49ers	jerseys
cornell	tulane	orientation	packers	nfl	tshirts
tulane	nyu	haven	nfl	packers	seattle
tufts	tufts	graduate	steelers	steelers	hats

similar to the query term "Albuquerque". Intuitively, a document about "Tucson"—which is typically similar to "Albuquerque"—is unlikely to satisfy the user intent. The discussion in §3.2 on how input features influence the notion of similarity in the learnt vector space is relevant here.

Models, such as LSA (Deerwester *et al.*, 1990) and Paragraph2vec (Le and Mikolov, 2014), that consider term-document pairs generally capture topical similarities in the learnt vector space. On the other hand, word2vec (Mikolov *et al.*, 2013b) and GloVe (Pennington *et al.*, 2014) embeddings may incorporate a mixture of topical and typical notions of relatedness. The inter-term relationships modelled in these latent spaces may be closer to type-based similarities when trained with short window sizes or on short text, such as on keyword queries (Levy and Goldberg, 2014; Mitra *et al.*, 2016a).

In §3.4, we note that the word2vec model learns two different embeddings—IN and OUT—corresponding to the input and the output terms. In retrieval, if a query contains a term t_i then—in addition to the frequency of occurrences of t_i in the document—we may also consider the presence of a different term t_j in the document to be a

supporting evidence of relevance if the pair of terms $\langle t_i, t_j \rangle$ frequently co-occurs in the collection. As shown in Equation (3.5), in the skip-gram model this probability of co-occurrence $p(t_j|t_i)$ is proportional to $(W_{out}\vec{v}_{t_j})^\intercal (W_{in}\vec{v}_{t_i})$—*i.e.*, the dot product between the IN embeddings of t_i and the OUT embeddings of t_j. Therefore, Nalisnick *et al.* (2016) point out that when using word2vec embeddings for estimating the relevance of a document to a query, it is more appropriate to compute the IN-OUT similarity between the query and the document terms. In other words, the query terms should be represented using the IN embeddings and the document terms using the OUT embeddings. Table 4.1 highlights the difference between IN-IN or IN-OUT similarities between terms.

The proposed *Dual Embedding Space Model* (DESM)[1] (Nalisnick *et al.*, 2016; Mitra *et al.*, 2016a) estimates the query-document relevance as follows,

$$DESM_{in-out}(q, d) = \frac{1}{|q|} \sum_{t_q \in q} \frac{\vec{v}_{t_q,in}^\intercal \vec{v}_{d,out}}{\|\vec{v}_{t_q,in}\| \|\vec{v}_{d,out}\|} \qquad (4.4)$$

$$\vec{v}_{d,out} = \frac{1}{|d|} \sum_{t_d \in d} \frac{\vec{v}_{t_d,out}}{\|\vec{v}_{t_d,out}\|} \qquad (4.5)$$

An alternative to representing queries and documents as an aggregate of their term embeddings is to incorporate the term representations into existing IR models, such as the ones we discussed in §2.5. Zuccon *et al.* (2015) proposed the *Neural Translation Language Model* (NTLM) that uses the similarity between term embeddings as a measure for term-term translation probability $p(t_q|t_d)$ in Equation (2.17).

$$p(t_q|t_d) = \frac{cos(\vec{v}_{t_q}, \vec{v}_{t_d})}{\sum_{t \in T} cos(\vec{v}_t, \vec{v}_{t_d})} \qquad (4.6)$$

On similar lines, Ganguly *et al.* (2015) proposed the *Generalized Language Model* (GLM) which extends the Language Model based

---

[1]The dual term embeddings trained on Bing queries is available for download at https://www.microsoft.com/en-us/download/details.aspx?id=52597

approach in Equation (2.15) to,

$$p(d|q) = \prod_{t_q \in q} \left(\lambda \frac{tf(t_q, d)}{|d|} + \alpha \frac{\sum_{t_d \in d} \left(sim(\vec{v}_{t_q}, \vec{v}_{t_d}) \cdot tf(t_d, d) \right)}{\sum_{t_{d_1} \in d} \sum_{t_{d_2} \in d} sim(\vec{v}_{t_{d_1}}, \vec{v}_{t_{d_2}}) \cdot |d|^2} \right.$$

$$+ \beta \frac{\sum_{\bar{t} \in N_t} \left(sim(\vec{v}_{t_q}, \vec{v}_{\bar{t}}) \cdot \sum_{\bar{d} \in D} tf(\bar{t}, \bar{d}) \right)}{\sum_{t_{d_1} \in N_t} \sum_{t_{d_2} \in N_t} sim(\vec{v}_{t_{d_1}}, \vec{v}_{t_{d_2}}) \cdot \sum_{\bar{d} \in D} |\bar{d}| \cdot |N_t|}$$

$$\left. + (1 - \alpha - \beta - \lambda) \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|} \right) \tag{4.7}$$

where, N_t is the set of nearest-neighbours of term t. Ai *et al.* (2016a) incorporate paragraph vectors (Le and Mikolov, 2014) into the query-likelihood model (Ponte and Croft, 1998).

Another approach, based on the Earth Mover's Distance (EMD) (Rubner *et al.*, 1998), involves estimating similarity between pairs of documents by computing the minimum distance in the embedding space that each term in the first document needs to travel to reach the terms in the second document. This measure, commonly referred to as the *Word Mover's Distance* (WMD), was originally proposed by Wan *et al.* (Wan and Peng, 2005; Wan, 2007), but used WordNet and topic categories instead of embeddings for defining the distance between terms. Term embeddings were later incorporated into the model by Kusner *et al.* (Kusner *et al.*, 2015; Huang *et al.*, 2016). Finally, Guo *et al.* (2016b) incorporated similar notion of distance into the *Non-linear Word Transportation* (NWT) model that estimates relevance between a query and a document. The NWT model involves solving the following constrained optimization problem,

$$\max \quad \sum_{t_q \in q} \log \left(\sum_{t_d \in u(d)} f(t_q, t_d) \cdot max(cos(\vec{v}_{t_q}, \vec{v}_{t_d}), 0)^{idf(t_q) + b} \right) \tag{4.8}$$

$$\text{subject to} \quad f(t_q, t_d) \geq 0, \quad \forall t_q \in q, t_d \in d \tag{4.9}$$

$$\text{and} \quad \sum_{t_q \in q} f(t_q, t_d) = \frac{tf(t_d) + \mu \frac{\sum_{\bar{d} \in D} tf(t_q, \bar{d})}{\sum_{\bar{d} \in D} |\bar{d}|}}{|d| + \mu}, \quad \forall t_d \in d \quad (4.10)$$

$$\text{where,} \quad idf(t) = \frac{|D| - df(t) + 0.5}{df(t) + 0.5} \quad (4.11)$$

$u(d)$ is the set of all unique terms in document d, and b is a constant.

Another term-alignment based distance metric was proposed by Kenter and Rijke (2015) for computing short-text similarity. The design of the *saliency-weighted semantic network* (SWSN) is motivated by the BM25 (Robertson and Zaragoza, 2009) formulation.

$$swsn(s_l, s_s) = \sum_{t_l \in s_l} idf(t_l) \cdot \frac{sem(t_l, s_s) \cdot (k_1 + 1)}{sem(t_l, s_s) + k_1 \cdot \left(1 - b + b \cdot \frac{|s_s|}{avgsl}\right)}$$
$$(4.12)$$

$$\text{where,} \quad sem(t, s) = \max_{\bar{t} \in s} cos(\vec{v}_t, \vec{v}_{\bar{t}}) \quad (4.13)$$

Here s_s is the shorter of the two sentences to be compared, and s_l the longer sentence.

Figure 4.2 highlights the distinct strengths and weaknesses of matching using local and distributed representations of terms for retrieval. For the query "Cambridge", a local representation (or exact matching) based model can easily distinguish between the passage on Cambridge (Figure 4.2a) and the one on Oxford (Figure 4.2b). However, the model is easily duped by a non-relevant passage that has been artificially injected with the term "Cambridge" (Figure 4.2a). The embedding space based matching, on the other hand, can spot that the other terms in the passage provide clear indication that the passage is not *about* a city, but fails to realize that the passage about Oxford (Figure 4.2b) is inappropriate for the same query.

Embedding based models often perform poorly when the retrieval is performed over the full document collection (Mitra *et al.*, 2016a). However, as seen in the example of Figure 4.2, the errors made by embedding based models and exact matching models may be different— and the combination of the two is often preferred (Hofmann, 1999;

the city of cambridge is a university city and the county town of cambridgeshire , england . it lies in east anglia , on the river cam , about 50 miles (80 km) north of london . according to the united kingdom census 2011 , its population was _ (including _ students) . this makes cambridge the second largest city in cambridgeshire after peterborough , and the 54th largest in the united kingdom . there is archaeological evidence of settlement in the area during the bronze age and roman times ; under viking rule cambridge became an important trading centre . the first town charters were granted in the 12th century , although city status was not conferred until 1951 .

(a) Passage about the city of Cambridge

oxford is a city in the south east region of england and the county town of oxfordshire . with a population of _ it is the 52nd largest city in the united kingdom , and one of the fastest growing and most ethnically diverse . oxford has a broad economic base . its industries include motor manufacturing , education , publishing and a large number of information technology and _ businesses , some being academic offshoots . the city is known worldwide as the home of the university of oxford , the oldest university in the _ world . buildings in oxford demonstrate examples of every english architectural period since the arrival of the saxons , including the _ radcliffe camera . oxford is known as the city of dreaming spires , a term coined by poet matthew arnold .

(b) Passage about the city of Oxford

Figure 4.2: A visualization of IN-OUT similarities between terms in different passages with the query term "Cambridge". The visualization reveals that, besides the term "Cambridge", many other terms in the passages about both Cambridge and Oxford have high similarity to the query term. The passage (c) is adapted from a passage on giraffes by replacing all the occurrences of the term "giraffe" with "cambridge". However, none of the other terms in (c) are found to be relevant to the query term. An embedding based approach may be able to determine that passage (c) is non-relevant to the query "Cambridge", but fail to realize that passage (b) is also non-relevant. A term counting-based model, on the other hand, can easily identify that passage (b) is non-relevant but may rank passage (c) incorrectly high.

Ganguly *et al.*, 2015; Mitra *et al.*, 2016a; Ai *et al.*, 2016a). Another technique is to use the embedding based model to re-rank only a subset of the documents retrieved by a different—generally an exact matching based—IR model. The chaining of different IR models where each

the cambridge (giraffa camelopardalis) is an african _ ungulate mammal , the tallest living terrestrial animal and the largest ruminant . its species name refers to its _ shape and its _ colouring . its chief distinguishing characteristics are its extremely long neck and legs , its _ _ , and its distinctive coat patterns . it is classified under the family _ , along with its closest extant relative , the okapi . the nine subspecies are distinguished by their coat patterns . the scattered range of giraffes extends from chad in the north to south africa in the south , and from niger in the west to somalia in the east . giraffes usually inhabit savannas , grasslands , and open woodlands .

(a) Passage about giraffes, but 'giraffe' is replaced by 'Cambridge'

Figure 4.2: Continued.

successive model re-ranks a smaller number of candidate documents is called *Telescoping* (Matveeva *et al.*, 2006). Telescoping evaluations are common in the neural IR literature (Mitra *et al.*, 2016a; Huang *et al.*, 2013; Shen *et al.*, 2014a; Guo *et al.*, 2016a; Mitra *et al.*, 2017a) and the results are representative of performances of these models on re-ranking tasks. However, as Mitra *et al.* (2016a) demonstrate, good performances on re-ranking tasks may not be indicative how the model would perform if the retrieval involves larger document collections.

> Embedding based models often make different errors than exact matching models, and the combination of the two may be more effective.

4.2   Query expansion

Instead of comparing the query and the document directly in the embedding space, an alternative approach is to use term embeddings to find good expansion candidates from a global vocabulary, and then retrieving documents using the expanded query. Different functions (Diaz *et al.*, 2016; Roy *et al.*, 2016; Zamani and Croft, 2016a) have been
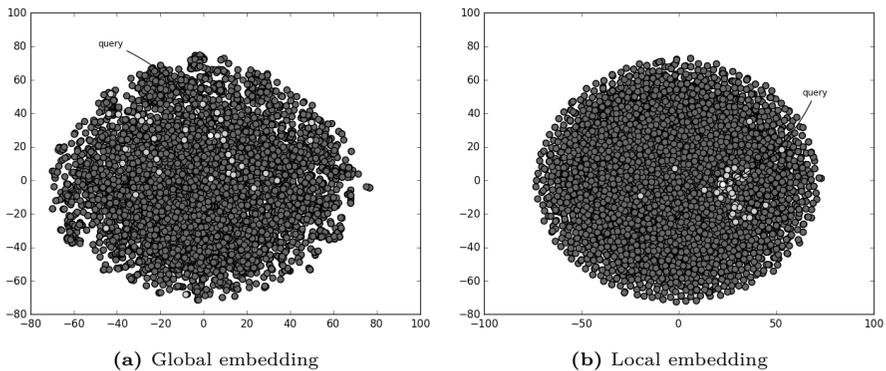
(a) Global embedding   **(b)** Local embedding

Figure 4.3: A two-dimensional visualization of term embeddings when the vector space is trained on a (a) global corpus and a (b) query-specific corpus, respectively. The grey circles represent individual terms in the vocabulary. The white circle represents the query "ocean remote sensing" as the centroid of the embeddings of the individual query terms, and the light grey circles correspond to good expansion terms for this query. When the representations are query-specific then the meaning of the terms are better disambiguated, and more likely to result in the selection of good expansion terms.

proposed for estimating the relevance of candidate terms to the query— all of them involves comparing the candidate term individually to every query term using their vector representations, and then aggregating the scores. For example, (Diaz *et al.*, 2016; Roy *et al.*, 2016) estimate the relevance of candidate term t_c as,

$$score(t_c, q) = \frac{1}{|q|} \sum_{t_q \in q} cos(\vec{v}_{t_c}, \vec{v}_{t_q}) \qquad (4.14)$$

Term embedding based query expansion on its own performs worse than pseudo-relevance feedback (Roy *et al.*, 2016). But like the models in the previous section, shows better performances when used in combination with PRF (Zamani and Croft, 2016a).

Diaz *et al.* (2016) explored the idea of query-specific term embeddings and found that they are more effective in identifying good expansion terms than a global representation (see Figure 4.3). The local model proposed by Diaz *et al.* (2016) incorporate relevance feedback in the process of learning the term embeddings—a set of documents is retrieved for the query and a query-specific term embedding model is trained.

This *local* embedding model is then employed for identifying expansion candidates for the query for a second round of document retrieval.

✎ Query-specific local analysis is more effective for query expansion than approaches that perform global analysis.

Term embeddings have also been explored for re-weighting query terms (Zheng and Callan, 2015) and finding relevant query re-writes (Grbovic *et al.*, 2015b), as well as in the context of other IR tasks such as crosslingual retrieval (Vulić and Moens, 2015) and entity retrieval (Van Gysel *et al.*, 2016a; Van Gysel *et al.*, 2016b). In §7, we will discuss neural network models with deeper architectures and their applications to retrieval.

5

Supervised learning to rank

Learning to rank (LTR) for IR uses training data $rel_q(d)$, such as human relevance labels and click data, to train towards an IR objective. This is distinct from the models in the previous section, which do not rely on IR data for ranking or for learning term representations. LTR models represent a rankable item—*e.g.*, a query-document pair—as a feature vector $\vec{x} \in \mathbb{R}^n$. The ranking model $f : \vec{x} \to \mathbb{R}$ is trained to map the vector to a real-valued score such that for a given query more relevant documents are scored higher and some chosen rank-based metric is maximized. The model training is said to be end-to-end if the parameters of f are learned all at once rather than in parts, and if the vector $\vec{x}$ contains simple features rather than models. Liu (2009) categorizes the different LTR approaches based on their training objectives.

- In the *pointwise approach*, the relevance information $rel_q(d)$ is in the form of a numerical value associated with every query-document pair with input vector $\vec{x}_{q,d}$. The numerical relevance label can be derived from binary or graded relevance judgments or from implicit user feedback, such as a clickthrough rate. A

55

regression model is typically trained on the data to predict the numerical value $rel_q(d)$ given $\vec{x}_{q,d}$.

- In the *pairwise approach*, the relevance information is in the form of preferences between pairs of documents with respect to individual queries (*e.g.*, $d_i \succ_q d_j$). The ranking problem in this case reduces to that of a binary classification to predict the more relevant document.

- Finally, the *listwise approach* involves directly optimizing for a rank-based metric such as NDCG—which is more challenging because these metrics are often not continuous (and hence not differentiable) with respect to the model parameters.

Many machine learning models—including support vector machines (Yue *et al.*, 2007), neural networks (Burges *et al.*, 2005), and boosted decision trees (Wu *et al.*, 2010)—have been employed over the years for the LTR task, and a correspondingly large number of different loss functions have been explored.

✏️ In learning to rank, direct supervision is used to optimize the model for a ranking task.

5.1 Input features

Traditional LTR models employ hand-crafted features (Liu, 2009) for representing query-document pairs in $\vec{x}$. The design of these features typically encodes key IR insights and belong to one of the three categories.

- *Query-independent* or *static* features (*e.g.*, incoming link count and document length)

- *Query-dependent* or *dynamic* features (*e.g.*, BM25)

- *Query-level* features (*e.g.*, query length)

In contrast, in recently proposed neural LTR models the deep architecture is responsible for feature learning[1] from simple vector representations of the input which may resemble the schemes described in §6.1 (*e.g.*, (Huang *et al.*, 2013)) or the interaction-based representations that we discuss later in §7.3 (*e.g.*, (Mitra *et al.*, 2017a; Pang *et al.*, 2016a)). These features, learnt from the query and document texts, can be combined with other features that may not be possible to infer from the content, such as document popularity (Shan *et al.*, 2016).

5.2 Loss functions

In ad-hoc retrieval, the LTR model needs to rank the documents in a collection D in response to a query. When training a neural model for this task, the ideal ranking of documents for a query q from the training dataset can be determined based on the relevance labels $rel_q(d)$ associated with each document $d \in D$. In the pointwise approach, the neural model is trained to directly estimate $rel_q(d)$, which can be a numeric value or a categorical label.

Regression loss   Given $\vec{x}_{q,d}$, the task of estimating the relevance label $rel_q(d)$ can be cast as a regression problem, and a standard loss function—such as the *square loss*—can be employed.

$$\mathcal{L}_{\text{squared}} = \|rel_q(d) - s(\vec{x}_{q,d})\|^2 \tag{5.1}$$

where, $s(\vec{x}_{q,d})$ is the score predicted by the model and $rel_q(d)$ can either be the value of the relevance label (Cossock and Zhang, 2006) or the one-hot representation when the label is categorical (Fuhr, 1989).

Classification loss   When the relevance labels in the training data are categorical, it makes more sense to treat the label prediction problem as

---

[1] In the literature, when the model is responsible for feature learning the task is sometimes categorized as "learning to match" (Li, Xu, *et al.*, 2014; Onal *et al.*, 2018). However, from a machine learning viewpoint, this distinction between whether $\vec{x}$ is a vector of hand-engineered features or a vector encoding of query-document text makes little difference to the LTR formulation described here. We, therefore, avoid making this distinction in favor of a more general definition.

a multiclass classification. The neural model under this setting, estimates the probability of a label y given $\vec{x}_{q,d}$. The probability of the correct label $y_{q,d}$ $(= rel_q(d))$ can be obtained by the softmax function,

$$p(y_{q,d}|q,d) = p(y_{q,d}|\vec{x}_{q,d}) = \frac{e^{\gamma \cdot s(\vec{x}_{q,d}, y_{q,d})}}{\sum_{y \in Y} e^{\gamma \cdot s(\vec{x}_{q,d}, y)}} \tag{5.2}$$

The softmax function normalizes the score of the correct label against the set of all possible labels Y. The *cross-entropy* loss can then be applied (Li *et al.*, 2008) as follows,

$$\mathcal{L}_{\text{classification}} = -\log\left(p(y_{q,d}|q,d)\right) = -\log\left(\frac{e^{\gamma \cdot s(\vec{x}_{q,d}, y_{q,d})}}{\sum_{y \in Y} e^{\gamma \cdot s(\vec{x}_{q,d}, y)}}\right) \tag{5.3}$$

However, a ranking model does not need to estimate the true relevance label accurately as long as it ranks the relevant documents D^+ over all the other candidates in D. Typically, only a few documents from D are relevant to q. If we assume a binary notion of relevance, then the problem is similar to multi-label classification—or, multiclass classification if we assume a single relevant document d^+ per query—where the candidate documents are the classes. Next, we discuss loss functions for LTR models that tries to predict the relevant document by maximizing $p(d^+|q)$. Note that this is different from the classification loss in Equation (5.3) which maximizes $p(y_{q,d}|q,d)$.

Contrastive loss   In representation learning models, a relevant document should be closer to the query representation than a non-relevant document. The contrastive loss (Chopra *et al.*, 2005; Hadsell *et al.*, 2006)—common in image retrieval—learns the model parameters by minimizing the distance between a relevant pair, while increasing the distance between dissimilar items.

$$\mathcal{L}_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot \mathcal{L}_{pos}(\text{dist}_{q,d}) \tag{5.4}$$
$$+ (1 - y_{q,d}) \cdot \mathcal{L}_{neg}(\text{dist}_{q,d}) \tag{5.5}$$

Contrastive loss assumes that the relevance label $y_{q,d} \in \{0, 1\}$ is binary. For each training sample, either $\mathcal{L}_{pos}$ or $\mathcal{L}_{neg}$ is applied over the

distance $\text{dist}_{q,d}$ as predicted by the model. In particular, Hadsell *et al.* (2006) use the following formulation of this loss function.

$$\mathcal{L}_{\text{Contrastive}}(q, d, y_{q,d}) = y_{q,d} \cdot \frac{1}{2} \left(\max(0, m - \text{dist}_{q,d}) \right)^2 \tag{5.6}$$

$$+ (1 - y_{q,d}) \cdot (\text{dist}_{q,d})^2 \tag{5.7}$$

where, m is a margin.

Cross-Entropy loss over documents   The probability of ranking d^+ over all the other documents in the collection D is given by the softmax function,

$$p(d^+|q) = \frac{e^{\gamma \cdot s\left(q, d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \tag{5.8}$$

The cross-entropy (CE) loss then maximizes the difference between scores generated by the model for relevant and less relevant documents.

$$\mathcal{L}_{\text{CE}}(q, d^+, D) = -\log \left(p(d^+|q) \right) \tag{5.9}$$

$$= -\log \left(\frac{e^{\gamma \cdot s\left(q, d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \right) \tag{5.10}$$

However, when D is the full collection then computing the softmax (*i.e.* the denominator in Equation (5.10)) is prohibitively expensive. Coincidentally, the CE loss is also useful for non-IR tasks, such as language modelling (Bengio *et al.*, 2003; Mikolov *et al.*, 2013a), where the model needs to predict a single term from a large vocabulary given its neighbours as input. Several different approaches have been proposed in the LM literature to address this computational complexity that is relevant to our discussion. We briefly describe some of these strategies here.

Hierarchical softmax   Instead of computing $p(d^+|q)$ directly, Goodman (2001) groups the candidates D into a set of classes C, and then predicts

the correct class c^+ given q followed by predicting d^+ given $\langle c^+, q \rangle$.

$$p(d^+|q) = p(d^+|c^+, x) \cdot p(c^+|q) \tag{5.11}$$

The computational cost in this modified approach is a function of $|C| + |c^+|$ which is typically much smaller than $|D|$. Further computational efficiency can be achieved by employing a hierarchy of such classes (Morin and Bengio, 2005; Mnih and Hinton, 2009). The hierarchy of classes is typically based on either similarity between candidates (Brown *et al.*, 1992; Le *et al.*, 2011; Mikolov *et al.*, 2013a), or frequency binning (Mikolov *et al.*, 2011). Zweig and Makarychev (2013) and Grave *et al.* (2016) have explored strategies for building the hierarchy that directly minimizes the computational complexity.

✎ Computing the cross-entropy loss with a softmax is costly because the softmax normalization involves scoring every candidate in the collection using the trained model. Alternatives include applying hierarchical categories over candidates or approximating the loss function by considering only a sample of candidates from the collection.

Importance sampling (IS)   An alternative to computing the exact softmax, is to approximately estimate it using sampling based approaches. Note, that we can re-write Equation (5.10) as follows,

$$\mathcal{L}_{\text{CE}}(q, d^+, D) = -\log \left(\frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \right) \tag{5.12}$$

$$= -\gamma \cdot s(q, d^+) + \log \sum_{d \in D} e^{\gamma \cdot s(q, d)} \tag{5.13}$$

To train a neural model using back-propagation, we need to compute the gradient ∇_θ of the loss $\mathcal{L}_{\text{CE}}$ with respect to the model parameters

θ,

$$\nabla_\theta \mathcal{L}_{\text{CE}}(q, d^+, Y) = -\gamma \nabla_\theta \cdot s(q, d^+) + \nabla_\theta \log \sum_{d \in D} e^{\gamma \cdot s(q,d)} \qquad (5.14)$$

$$= -\gamma \nabla_\theta \cdot s(q, d^+) + \frac{\nabla_\theta \sum_{d \in D} e^{\gamma \cdot s(q,d)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \qquad (5.15)$$

$$= -\gamma \nabla_\theta \cdot s(q, d^+) + \frac{\sum_{d \in D} \nabla_\theta e^{\gamma \cdot s(q,d)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \qquad (5.16)$$

$$= -\gamma \nabla_\theta \cdot s(q, d^+) + \frac{\sum_{d \in D} \gamma \cdot e^{\gamma \cdot s(q,d)} \nabla_\theta s(q, d)}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \qquad (5.17)$$

$$= -\gamma \nabla_\theta \cdot s(q, d^+) + \gamma \sum_{d \in D} \frac{e^{\gamma \cdot s(q,d)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \nabla_\theta s(q, d) \qquad (5.18)$$

$$= -\gamma \nabla_\theta \cdot s(q, d^+) + \gamma \sum_{d \in D} p(d|q) \nabla_\theta s(q, d) \qquad (5.19)$$

As Senécal and Bengio (2003) point out, the first component of the gradient $\gamma \nabla_\theta s(q, d^+)$ is the positive reinforcement to the model for the correct candidate d^+ and the second component $\gamma \sum_{d \in D} p(d|q) \nabla_\theta s(q, d)$ is the negative reinforcement corresponding to all the other (incorrect) candidates. The key idea behind sampling based approaches is to estimate the second component without computing the costly sum over the whole candidate set. In IS (Bengio, Senécal, *et al.*, 2003; Jean *et al.*, 2014; Bengio and Senécal, 2008; Jozefowicz *et al.*, 2016), Monte-Carlo method is used to estimate the second component.

Noise Contrastive Estimation (NCE) In NCE (Gutmann and Hyvärinen, 2010; Mnih and Teh, 2012; Vaswani *et al.*, 2013), the task is modified to that of a binary classification. The model is trained to distinguish a sample drawn from a true distribution $p(d|q)$ from a sample drawn from a noisy distribution $\tilde{p}(d)$. The training data contains k noisy samples for every true sample. Let, $\mathcal{E}$ and $\bar{\mathcal{E}}$ indicate that a sample is drawn

from the true and the noisy distributions, respectively. Then,

$$p(\mathcal{E}|q, d) = \frac{p(d|q)}{p(d|q) + k \times \tilde{p}(d)} \tag{5.20}$$

$$p(\bar{\mathcal{E}}|q, d) = \frac{k \times \tilde{p}(d)}{p(d|q) + k \times \tilde{p}(d)} \tag{5.21}$$

We want our model to learn the true distribution $p(d|q)$. Remember, that according to our model,

$$p(d|q) = \frac{e^{\gamma \cdot s(q,d)}}{\sum_{\bar{d} \in D} e^{\gamma \cdot s(q,\bar{d})}} \tag{5.22}$$

$$= \frac{e^{\gamma \cdot s(q,d)}}{z(q)} \tag{5.23}$$

A key efficiency trick involves setting $z(q)$ to 1 (Mnih and Teh, 2012; Vaswani *et al.*, 2013; Zoph *et al.*, 2016). Therefore,

$$p(d|q) = e^{\gamma \cdot s(q,d)} \tag{5.24}$$

putting Equation (5.24) back in Equation (5.20) and (5.21).

$$p(\mathcal{E}|q, d) = \frac{e^{\gamma \cdot s(q,d)}}{e^{\gamma \cdot s(q,d)} + k \times \tilde{p}(d)} \tag{5.25}$$

$$p(\bar{\mathcal{E}}|q, d) = \frac{k \times \tilde{p}(d)}{e^{\gamma \cdot s(q,d)} + k \times \tilde{p}(d)} \tag{5.26}$$

Finally, the NCE loss is given by,

$$\mathcal{L}_{\text{NCE}} = -\sum_{\langle x,d^+ \rangle} \left(log \ p(\mathcal{E}|x, d^+) + \sum_{i=1}^{k} \log \ p(\bar{\mathcal{E}}|x, y_i^-) \right) \tag{5.27}$$

$$= -\sum_{\langle x,d^+ \rangle} \left(log \ \frac{e^{\gamma \cdot s(q,d^+)}}{e^{\gamma \cdot s(q,d^+)} + k \times \tilde{p}(d^+)} + \sum_{i=1}^{k} \log \ \frac{k \times \tilde{p}(y_i^-)}{e^{\gamma \cdot s(q,d_i^-)} + k \times \tilde{p}(y_i^-)} \right) \tag{5.28}$$

Note, that the outer summation iterates over all the positive $\langle x, d^+ \rangle$ pairs in the training data.

Negative sampling (NEG) Mikolov *et al.* (2013b) modify the NCE loss by replacing $k \times \tilde{p}(d)$ with 1 in Equation (5.25) and (5.26).

$$p(\mathcal{E}|q,d) = \frac{e^{\gamma \cdot s(q,d)}}{e^{\gamma \cdot s(q,d)} + 1} \tag{5.29}$$

$$= \frac{1}{1 + e^{-\gamma \cdot s(q,d)}} \tag{5.30}$$

$$p(\bar{\mathcal{E}}|q,d) = \frac{1}{1 + e^{\gamma \cdot s(q,d)}} \tag{5.31}$$

which changes the NCE loss to the NEG loss.

$$\mathcal{L}_{\text{NEG}} = -\sum_{\langle x,d^+ \rangle} \left(log \ \frac{1}{1 + e^{-\gamma \cdot s(q,d^+)}} + \sum_{i=1}^{k} \log \ \frac{1}{1 + e^{\gamma \cdot s(q,d_i^-)}} \right) \tag{5.32}$$

BlackOut Related to both IS and NCE, is BlackOut (Ji *et al.*, 2015). It is an extension of the DropOut (Srivastava *et al.*, 2014) method that is often employed to avoid over-fitting in neural models with large number of parameters. DropOut is typically applied to the input or hidden layers of the network and involves randomly dropping a subset of the neural units and their corresponding connections. BlackOut applies the same idea to the output layer of the network for efficiently computing the loss. We refer readers to (Ji *et al.*, 2015) for more rigorous discussions on the relationship between IS, NCE, and DropOut.

For document retrieval Huang *et al.* (2013) approximate the cross-entropy loss of Equation (5.10) by replacing D with D'—where, $D' = \{d^+\} \cup D^-$ and D^- is a fixed number of randomly sampled candidates. Mitra *et al.* (2017a) use a similar loss function but focus on the document re-ranking task where the neural model needs to distinguish the relevant documents from less relevant (but likely not completely non-relevant) candidates. Therefore, in their work the re-ranking model is trained with negative examples which comprise of documents retrieved by an existing IR system but manually judged as less relevant, instead of being sampled uniformly from the collection. IS, NCE, NEG, and these other sampling based approaches approximate the comparison with the full collection based on a sampled subset. For additional notes on these

approaches, we refer the readers to (Dyer, 2014; Chen *et al.*, 2015b; Ruder, 2016).

In a typical retrieval scenario, however, multiple documents may be relevant to the same query q, and the notion of relevance among this set of documents D^+ may be further graded. Some LTR approaches consider pairs of documents for the same query and minimize the average number of inversions in ranking—i.e., $d_i \succ_q d_j$ but d_j is ranked higher than d_i. The pairwise loss employed in these approaches has the following form (Chen *et al.*, 2009),

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \tag{5.33}$$

where, some possible choices for ϕ include,

- Hinge function $\phi(z) = \max(0, 1 - z)$ (Herbrich *et al.*, 2000; Schroff *et al.*, 2015)

- Exponential function $\phi(z) = e^{-z}$ (Freund *et al.*, 2003)

- Logistic function $\phi(z) = \log\left(1 + e^{-z}\right)$ (Burges *et al.*, 2005)

RankNet loss   RankNet (Burges *et al.*, 2005) is a *pairwise* loss function that has been a popular choice for training neural LTR models and was also for many years an industry favourite, such as at the commercial Web search engine Bing.[2] Under the RankNet loss, the model is trained on triples $\langle q, d_i, d_j \rangle$ consisting of a query q and a pair of documents d_i and d_j with different relevance labels—such that d_i is more relevant than d_j (*i.e.*, $d_i \succ_q d_j$)—and corresponding feature vectors $\langle \vec{x}_i, \vec{x}_j \rangle$. The model $f : \mathbb{R}^n \to \mathbb{R}$, typically a neural network but can also be any other machine learning model whose output is differentiable with respect to its parameters, computes the scores $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$, where ideally $s_i > s_j$. Given the scores $\langle s_i, s_j \rangle$, the probability that d_i would be ranked higher than d_j is given by,

$$p_{ij} \equiv p(s_i > s_j) \equiv \frac{1}{1 + e^{-\sigma(s_i - s_j)}} \tag{5.34}$$

---

[2]https://www.microsoft.com/en-us/research/blog/ranknet-a-ranking-retrospective/

where, the constant σ determines the shape of the sigmoid. During training, the probability of ranking d_i higher than d_j for q is maximized. Let $S_{ij} \in \{-1, 0, +1\}$ be the true preference label between d_i and d_j for the training sample— denoting d_i is less, equal, or more relevant than d_j, respectively. Then the desired probability of ranking d_i over d_j is given by $\bar{p}_{ij} = \frac{1}{2}(1 + S_{ij})$. The cross-entropy loss $\mathcal{L}$ between the desired probability $\bar{p}_{ij}$ and the predicted probability p_{ij} is given by,

$$\mathcal{L} = -\bar{p}_{ij}\log(p_{ij}) - (1 - \bar{p}_{ij})log(1 - p_{ij}) \tag{5.35}$$

$$= \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{-\sigma(s_i - s_j)}) \tag{5.36}$$

$$= \log(1 + e^{-\sigma(s_i - s_j)}) \quad \text{if, } d_i \underset{q}{\succ} d_j (S_{ij} = 1) \tag{5.37}$$

Note that $\mathcal{L}$ is differentiable with respect to the model output s_i and hence the model can be trained using gradient descent. We direct the interested reader to (Burges, 2010) for more detailed derivations for computing the gradients for RankNet.

Readers should note the obvious connection between the CE loss described previously and the RankNet loss. If in the denominator of Equation (5.10), we only sum over a pair of relevant and non-relevant documents then it reduces to the logistic-loss function of RankNet described in Equation (5.37). So, at the level of a single training sample, the key distinction between the two is whether we compare the relevant document to a single less relevant candidate or the full collection. However, in case of RankNet, it is important to consider how the pairs are sampled as the training is influenced by their distribution.

The key limitation of pairwise objective functions is that the rank inversion of any pair of documents is considered equally harmful. This is, however, generally untrue for most IR metrics where a significantly large penalty is associated with inversions at the top rank positions. For example, consider two different result lists for the same query—result list A ranks two relevant documents at position one and 50, while result list B ranks the same two relevant documents at positions three and 40. While the result set A has more rank inversions compared to result set B (48 vs. 40), it would fare better on typical IR metrics, such as NDCG.

Therefore, to optimize for a rank-based metric we need to incorporate listwise objectives—that are sensitive to these differences—in our model training. However, the rank-based metrics are generally non-continuous and non-differentiable, which makes them difficult to incorporate in the loss function.

LambdaRank loss   Burges *et al.* (2006) make two key observations: (i) the gradient should be bigger for pairs of documents that produce a bigger impact in NDCG by swapping positions, and (ii) to train a model we don't need the costs themselves, only the gradients (of the costs w.r.t model scores). This leads to the LambdaRank loss which weights the gradients from the RankNet loss by the NDCG delta that would result from swapping the rank position of the pair of documents.

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta NDCG| \qquad (5.38)$$

This formulation of LambdaRank can optimize directly for NDCG (Donmez *et al.*, 2009; Yue and Burges, 2007), and any other IR measure by incorporating the corresponding delta change in Equation (5.38).

ListNet and ListMLE loss   The probability of observing a particular rank order can be estimated from the individual document scores using different models (Luce, 1959; Plackett, 1975; Mallows, 1957). For example, according to the Luce model (Luce, 1959), given four items $\{d_1, d_2, d_3, d_4\}$ the probability of observing a particular rank-order, say $[d_2, d_1, d_4, d_3]$, is given by:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \times \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)}$$
$$\times \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)}$$
$$(5.39)$$

where, π is a particular permutation and ϕ is a transformation (e.g., linear, exponential, or sigmoid) over the score s_i corresponding to item d_i. Using this model, we can compute the probability distribution over all possible permutations based on the model scores and the ground

truth labels. The K-L divergence between these two distributions gives us the ListNet loss (Cao *et al.*, 2007).

However, computing the probability distribution over all possible permutations is computationally expensive, even when restricted to only the top-K items. The ListMLE loss (Xia *et al.*, 2008) instead computes the probability of the ideal permutation based on the ground truth. However, with categorical labels more than one ideal permutation may be possible which should be handled appropriately.

Many of the challenges discussed in this section are common to both retrieval tasks as well as multiclass and multilabel classification with extremely large number of classes—often referred to as *extreme classification* (Bhatia *et al.*, 2016; Varma and Cisse, 2015; Cisse *et al.*, 2016). Ad-hoc retrieval can be posed as an extreme classification task under a binary notion of relevance and a fixed collection constraint. New loss functions (*e.g.* the spherical loss family (Vincent *et al.*, 2016; Brébisson and Vincent, 2015; Brébisson and Vincent, 2016)) have been explored for these large scale classification tasks which may be relevant for neural retrieval research. The problem of learning from sparse biased labels (Joachims *et al.*, 2017; Jain *et al.*, 2016) is also an important challenge in these frameworks. Finally, deep neural models for LTR with significantly large number of parameters may require large amount of training data for supervised learning. Alternative training schemes—*e.g.*, using weak supervision signals (Dehghani *et al.*, 2017b; Dehghani *et al.*, 2017a) or adversarial learning (Wang *et al.*, 2017; Cohen *et al.*, 2018)—are emerging.

6

Deep neural networks

Deep neural network models consist of chains of tensor operations. The tensor operation can range from parameterized linear transformations (*e.g.*, multiplication with a weight matrix, or the addition of a bias vector) to elementwise application of non-linear functions, such as *tanh* or *rectified linear units* (ReLU) (Hahnloser *et al.*, 2000; Nair and Hinton, 2010; Jarrett *et al.*, 2009). Figure 6.1 shows a simple *feed-forward* neural network with *fully-connected* layers. For an input vector $\vec{x}$, the model produces the output $\vec{y}$ as follows,

$$\vec{y} = tanh(W_2 \cdot tanh(W_1 \cdot \vec{x} + \vec{b}_1) + \vec{b}_2) \qquad (6.1)$$

The model training involves tuning the parameters W_1, $\vec{b}_1$, W_2, and $\vec{b}_2$ to minimize the loss between the expected output and the output predicted by the final layer. The parameters are usually trained discriminatively using backpropagation (Schmidhuber, 2015; Bengio, 2009; Hecht-Nielsen *et al.*, 1988). During forward-pass each layer generates an output conditioned on its input, and during backward pass each layer computes the error gradient with respect to its parameters and its inputs.

The design of a DNN typically involves many choices of architectures and hyper-parameters. Neural networks with as few a single hidden
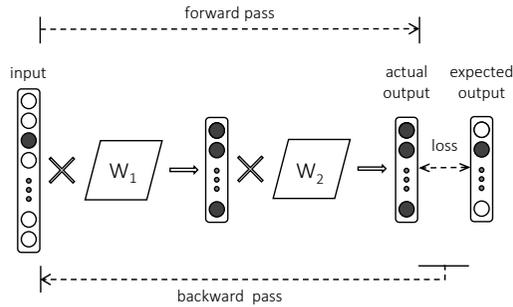
(a) A neural network with a single hidden layer
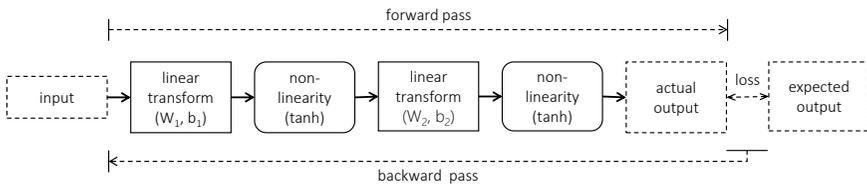
(b) The same neural network viewed as a chain of computational steps

Figure 6.1: Two different visualizations of a feed-forward neural network with a single hidden layer. In (a), the addition of the bias vector and the non-linearity function is implicit. Figure (b) shows the same network but as a sequence of computational nodes. Most neural network toolkits implement a set of standard computational nodes that can be connected to build more sophisticated neural architectures.

layer—but with sufficient number of hidden nodes—can theoretically approximate any function (Hornik *et al.*, 1989). In practice, however, deeper architectures—sometimes with as many as 1000 layers (He *et al.*, 2016)—have been shown to perform significantly better than shallower networks. For readers who are less familiar with neural network models, we present a simple example in Figure 6.2 to illustrate how hidden layers enable these models to capture non-linear relationships. We direct readers to (Montufar *et al.*, 2014) for further discussions on how additional hidden layers help.

The rest of this section is dedicated to the discussion of input representations and standard architectures for deep neural models.

Input features				Hidden layers		Label
surface	kerberos	book	library	H_1	H_2	
1	0	1	0	1	0	✓
1	1	0	0	0	0	✗
0	1	0	1	0	1	✓
0	0	1	1	0	0	✗

Figure 6.2: Consider a toy binary classification task on a corpus of four short texts—"surface book", "kerberos library", "library book", and "kerberos surface"—where the model needs to predict if the text is related to computers. The first two texts—"Surface Book" and "kerberos library"—are positive under this classification, and the latter two negative. The input feature space consists of four binary features that indicate whether each of the four terms from the vocabulary is present in the text. The table shows that the specified classes are not linearly separable with respect to the input feature space. However, if we add couple of hidden nodes, as shown in the diagram, then the classes can be linearly separated with respect to the output of the hidden layer.

6.1 Input text representations

Neural models that learn representations of text take raw text as input. A key consideration is how the text should be represented at the input layer of the model. Figure 6.3 shows some of the common input representations of text.

Some neural models (Jozefowicz *et al.*, 2016; Graves, 2013; Sutskever *et al.*, 2011; Kim *et al.*, 2015) operate at the character-level. In these models, each character is typically represented by a one-hot vector. The vector dimensions—referred to as *channels*—in this case equals the number of allowed characters in the vocabulary. These models incorporate the least amount of prior knowledge about the language in the input representation—for example, these models are often required to learn about tokenization from scratch by treating space as just another character in the vocabulary. The representation of longer texts, such as sentences, can be derived by concatenating or summing the character-level vectors as shown in Figure 6.3a.

Input representation of text is typically in the form of one-hot representations—of characters, character n-grams, or terms—or pre-trained embeddings.

The input text can also be pre-tokenized into terms—where each term is represented by either a sparse vector or using pre-trained term embeddings (Figure 6.3d). Terms may have a one-hot (or local) representation where each term has an unique ID (Figure 6.3b), or the term vector can be derived by aggregating one-hot vectors of its constituting characters (or character n-graphs) as shown in Figure 6.3c. If pre-trained embeddings are used for term representation, then the embedding vectors can be further tuned during training or kept fixed.

Similar to character-level models, the term vectors are further aggregated (by concatenation or sum) to obtain the representation of longer chunks of text, such as sentences. While one-hot representations of terms (Figure 6.3b) are common in many NLP tasks, historically pre-trained embeddings (*e.g.*, (Pang *et al.*, 2016b; Hu *et al.*, 2014)) and character n-graph based representations (*e.g.*, (Huang *et al.*, 2013; Mitra *et al.*, 2017a)) are more commonplace in IR.

6.2 Standard architectures

In this section, we describe few standard neural architectures commonly used in IR. For broader overview of neural architectures and design patterns please refer to (Goodfellow *et al.*, 2016; LeCun *et al.*, 2015; Schmidhuber, 2015).

Shift-invariant neural operations Convolutional (LeCun *et al.*, 2004; Jarrett *et al.*, 2009; Krizhevsky *et al.*, 2012; LeCun *et al.*, 2010) and recurrent (Mikolov *et al.*, 2010; Graves *et al.*, 2009; Sak *et al.*, 2014; Hochreiter and Schmidhuber, 1997) architectures are commonplace in many deep learning applications. These neural operations are part of a
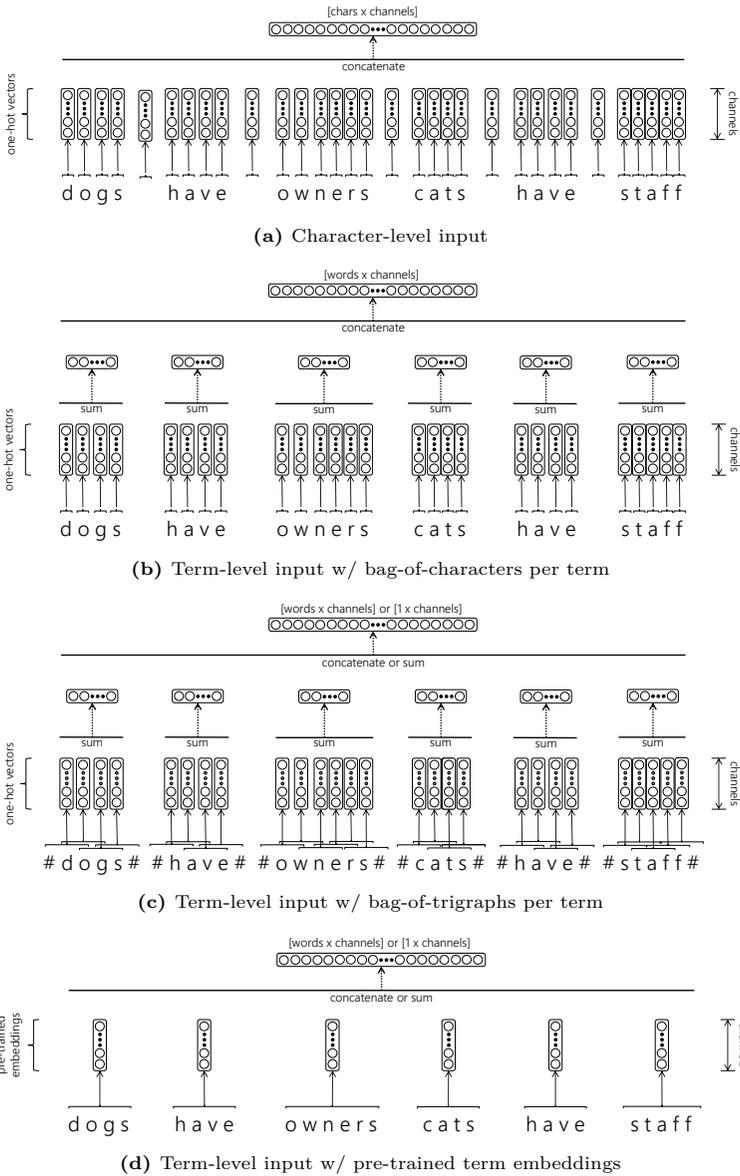
(a) Character-level input

(b) Term-level input w/ bag-of-characters per term

(c) Term-level input w/ bag-of-trigraphs per term

(d) Term-level input w/ pre-trained term embeddings

Figure 6.3: Examples of different representation strategies for text input to deep neural network models. The smallest granularity of representation can be a character or a term. The vector can be a sparse local representation, or a pre-trained embedding.

broader family of shift-invariant architectures. The key intuition behind these architectures stem from the natural regularities observable in most inputs. In vision, for example, the task of detecting a face should be invariant to whether the image is shifted, rotated, or scaled. Similarly, the meaning of an English sentence should, in most cases, stay consistent independent of which part of the document it appears in. Therefore, intuitively a neural model for object recognition or text understanding should not learn an independent logic for the same action applied to different parts of the input space. All shift-invariant neural operations fundamentally employ a window-based approach. A fixed size window moves over the input space with fixed stride in each step. A (typically parameterized) function—referred to as a *kernel*, or a *filter*, or a *cell*—is applied over each instance of the window. The parameters of the cell are shared across all the instances of the input window. The shared parameters not only imply a smaller number of total parameters in the model, but also more supervision per parameter per training sample due to the repeated application.

> All shift-invariant neural operations, including convolutional and recurrent layers, move a fixed size window over the input space with fixed stride. Each window is projected by a parameterized neural operation, or *cell*, often followed by an aggregation step, such as pooling.

   Figure 6.4a shows an example of a cell being applied on a sequence of terms—with a window size of three terms—in each step. A common cell implementation involves multiplying with a weight matrix—in which case the architecture in Figure 6.4a is referred as *convolutional*. An example of a cell without any parameters is *pooling*—which consists of aggregating (*e.g.*, by computing the max or the average per channel) over all the terms in the window. Note, that the length of the input
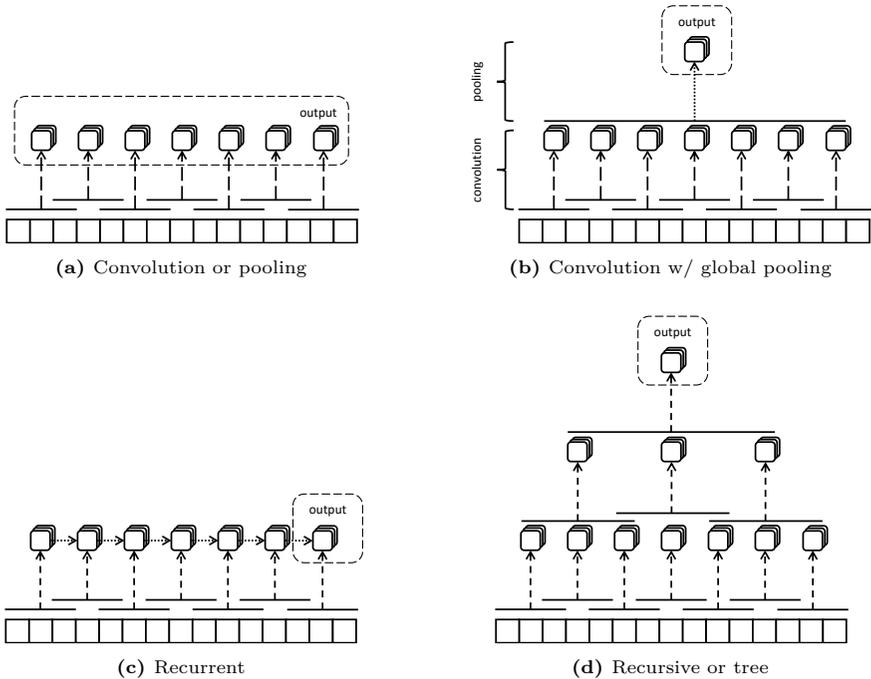
(a) Convolution or pooling

(b) Convolution w/ global pooling

(c) Recurrent

(d) Recursive or tree

Figure 6.4: Standard shift-invariant neural architectures including convolutional neural networks (CNN), recurrent neural networks (RNN), pooling layers, and tree-structured neural networks.

sequence can be variable in both cases and the length of the output of a convolutional (or pooling) layer is a function of the input length. Figure 6.4b shows an example of *global pooling*—where the window spans over the whole input—being applied on top of a convolutional layer. The global pooling strategy is common for generating a fixed size output from a variable length input.[1]

In convolution or pooling, each window is applied independently. In contrast, in the *recurrent* architecture of Figure 6.4c the cell not only

---

[1]It may be obvious, but worth pointing out, that a *global convolutional* layer is exactly the same as a fully-connected layer.

considers the input window but also the output of the previous instance of the cell as its input. Many different cell architectures have been explored for recurrent neural networks (RNN)—although Elman network (Elman, 1990), Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unit (GRU) (Chung *et al.*, 2014; Cho *et al.*, 2014) are commonly used. RNNs are popularly applied to sequences but can also be useful for two (and higher) dimensional inputs (Wan *et al.*, 2016).

One consideration when using convolutional or recurrent layers is how the window outputs are aggregated. Convolutional layers are typically followed by pooling or fully-connected layers that perform a global aggregation over all the window instances. While a fully-connected layer is aware of each window position, a global pooling layer is typically agnostic to it. However, unlike a fully-connected layer, a global max-pooling operation can be applied to a variable size input. Where a global aggregation strategy may be less appropriate (*e.g.*, long sequences), recurrent networks with memory (Weston *et al.*, 2014; Sukhbaatar *et al.*, 2015; Bordes *et al.*, 2015) and/or attention (Mnih *et al.*, 2014; Xu *et al.*, 2015; Luong *et al.*, 2015; Hermann *et al.*, 2015; Chorowski *et al.*, 2015) may be useful.

Finally, Figure 6.4c shows *tree-structured* (or *recursive*) neural networks (Goller and Kuchler, 1996; Socher *et al.*, 2011b; Bowman *et al.*, 2016; Tai *et al.*, 2015; Socher *et al.*, 2011a) where the same cell is applied at multiple levels in a tree-like hierarchical fashion.

Autoencoders The autoencoder architecture (Bengio *et al.*, 2007; Ranzato *et al.*, 2006; Bengio, 2009) is based on the *information bottleneck method* (Tishby *et al.*, 2000). The goal is to learn a compressed representation $\vec{x} \in \mathbb{R}^k$ of items from their higher-dimensional vector representations $\vec{v} \in \mathbb{R}^K$, such that $k \ll K$. The model has an hour-glass shape as shown in Figure 6.5a and is trained by feeding in the high-dimensional vector inputs and trying to re-construct the same representation at the output layer. The lower-dimensional middle layer forces the encoder part of the model to extract the *minimal sufficient statistics* of $\vec{v}$ into $\vec{x}$, such that the decoder part of the network can reconstruct the original input back from $\vec{x}$. The model is trained by minimizing the reconstruction
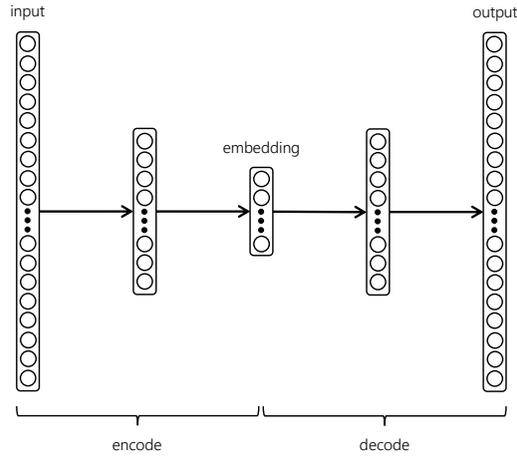
(a) Autoencoder
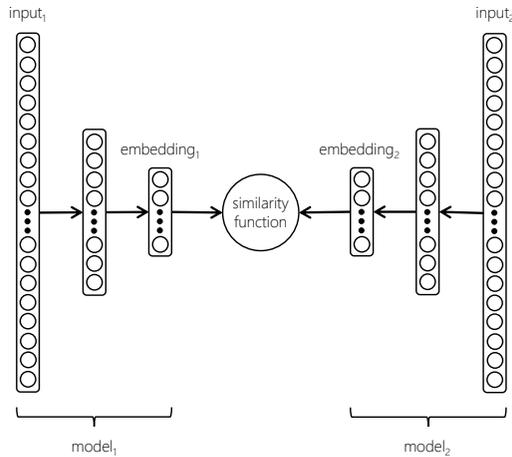
(b) Siamese network

Figure 6.5: Both (a) the autoencoder and (b) the Siamese network architectures are designed to learn compressed representations of inputs. In an autoencoder the embeddings are learnt by minimizing the self-reconstruction error, whereas a Siamese network focuses on retaining the information that is necessary for determining the similarity between a pair of items (say, a query and a document).

error between the input $\vec{v}$ and the actual output of the decoder $\vec{v'}$. The squared-loss is commonly employed.

$$\mathcal{L}_{autoencoder}(\vec{v}, \vec{v'}) = \|\vec{v} - \vec{v'}\|^2 \tag{6.2}$$

✎ Both autoencoders and Siamese networks learn compressed representations of the input. In autoencoders, the learnt representation is optimized for reducing reconstruction errors, whereas Siamese networks optimize to better discriminate similar pairs from dissimilar ones.

Siamese networks   Siamese networks were originally proposed for comparing fingerprints (Baldi and Chauvin, 1993) and signatures (Bromley *et al.*, 1993). Yih *et al.* (2011) later adapted the same architecture for comparing short texts. The Siamese network, as seen in Figure 6.5b, resembles the autoencoder architecture (if you squint hard enough)—but unlike the latter is trained on pairs of inputs $\langle input_1, input_2 \rangle$. The architecture consists of two models ($model_1$ and $model_2$) that project $input_1$ and $input_2$, respectively, to $\vec{v_1}$ and $\vec{v_2}$ in a common latent space. A pre-defined metric (*e.g.*, cosine similarity) is used to then compute the similarity between $\vec{v_1}$ and $\vec{v_2}$. The model parameters are optimized such that $\vec{v_1}$ and $\vec{v_2}$ are closer when the two inputs are expected to be similar, and further away otherwise.

One possible loss function is the logistic loss. If each training sample consist of a triple $\langle \vec{v_q}, \vec{v_{d1}}, \vec{v_{d2}} \rangle$, such that $sim(\vec{v_q}, \vec{v_{d1}})$ should be greater than $sim(\vec{v_q}, \vec{v_{d2}})$, then we minimize,

$$\mathcal{L}_{siamese}(\vec{v_q}, \vec{v_{d1}}, \vec{v_{d2}}) = \log\left(1 + e^{-\gamma(sim(\vec{v_q},\vec{v_{d1}})-sim(\vec{v_q},\vec{v_{d2}}))}\right) \tag{6.3}$$

where, γ is a constant that is often set to 10. Typically, both the models—$model_1$ and $model_2$—share identical architectures, but can also choose to share the same parameters. In image retrieval, the contrastive loss
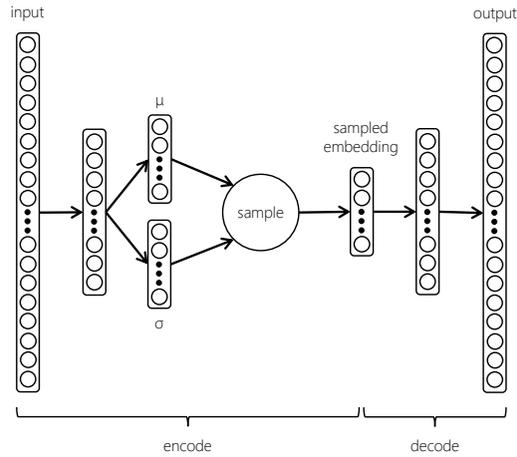
Figure 6.6: Instead of directly generating an encoded representation, variational autoencoders sample the latent vector from the generated vector of means μ and standard deviations σ. This local variation forces the model to learn a smoother and more continuous latent space.

(Chopra *et al.*, 2005; Hadsell *et al.*, 2006) is also used for training Siamese networks.

It is important to note that, unlike the autoencoder, the minimal sufficient statistics retained by a Siamese network is dictated by which information it deems important for determining the similarity between the paired items.

Variational autoencoders (VAE)   In Variational autoencoders (Kingma and Welling, 2014; Rezende *et al.*, 2014), the encoder part of the network generates two separate vectors—the vector of means μ and the vector of standard deviations σ. The latent representation $\vec{x}$ of the input is then generated by sampling a random variable x_i with mean μ_i and standard deviation σ_i along each of the k latent dimensions.

$$\vec{x} = [x_0 \sim N(\mu_0, \sigma_0^2), \ldots, x_i \sim N(\mu_i, \sigma_i^2), \ldots, x_{k-1} \sim N(\mu_{k-1}, \sigma_{k-1}^2)]$$
(6.4)

By sampling the latent representation, we expose the decoder to a certain degree of local variations in its input that should force the model to

learn a smoother continuous latent space. The VAE is trained by jointly minimizing the reconstruction loss—similar to vanilla autoencoders— and an additional component to the loss function which is the KL-divergence between the latent variable x_i and a unit gaussian.

$$\mathcal{L}_{\mathrm{VAE}} = \mathcal{L}_{reconstruction} + \mathcal{L}_{KL-divergence} \tag{6.5}$$

$$= \|\vec{v} - \vec{v'}\|^2 + \sum_{i}^{k} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1 \tag{6.6}$$

Without the $\mathcal{L}_{KL-divergence}$ component the model can learn very different μ for different classes of inputs and minimize the λ to be arbitrarily small such that the learnt latent space is no longer smooth or continuous. Readers should note that the sampling step is non-differentiable, but the model can be trained using the "reparameterization trick" proposed by Kingma and Welling (2014).

An important application of VAE is for the synthesis of new items (*e.g.*, images (Gregor *et al.*, 2015) or text (Bahuleyan *et al.*, 2017)) not observed in the training collection. Another popular class of techniques for synthesis includes the Generative Adversarial Networks.

Generative Adversarial Networks (GAN) Goodfellow *et al.* (2014) proposed a framework for training generative models under an adversarial setting. GANs typically consist of two separate neural networks—a *generator* network and a *discriminator* network. The goal of the generator network is to synthesize new (fake) items that mimic similar distributions as items that exist in the training collection. The goal of the discriminator network is to correctly distinguish between a true item and an item produced by the generator. The generator is trained to maximize the probability of the discriminator wrongly classifying the true and the generated item—which corresponds to a minimax two-player game.

6.3   Neural toolkits

In recent years, the advent of numerous flexible toolkits (Jia *et al.*, 2014; Yu *et al.*, n.d.; Abadi *et al.*, 2016; Collobert *et al.*, 2011a; Chen *et al.*, 2015a; Al-Rfou *et al.*, 2016; Tokui *et al.*, 2015; Neubig *et al.*, 2017) has had a catalytic influence on the area of neural networks. Most of the toolkits define a set of common neural operations that—like Lego[2] blocks—can be composed to build complex network architectures. Each instance of these neural operations or *computation nodes* can have associated learnable parameters that are updated during training, and these parameters can be shared between different parts of the network if necessary. Every computation node under this framework must implement the appropriate logic for,

- computing the output of the node given the input (forward-pass),

- computing the gradient of the loss with respect to the inputs, given the gradient of the loss with respect to the output (backward-pass),

- computing the gradient of the loss with respect to its parameters, given the gradient of the loss with respect to the output (backward-pass).

A deep neural network, such as the one in Figure 6.1 or ones with much more complex architectures (*e.g.*, (He *et al.*, 2016; Szegedy *et al.*, 2015; Larsson *et al.*, 2016)), can then be specified by chaining instances of these available computation nodes, and trained end-to-end on large datasets using backpropagation over GPUs or CPUs. In IR, various application interfaces (Van Gysel *et al.*, 2017a; Mitra *et al.*, 2017b) bind these neural toolkits with existing retrieval/indexing frameworks, such as Indri (Strohman *et al.*, 2005). Refer to (Shi *et al.*, 2016) for a comparison of different neural toolkits based on their speed of training using standard performance benchmarks.

---

[2]https://en.wikipedia.org/wiki/Lego

7

Deep neural networks for IR

Traditionally, deep neural network models have much larger number of learnable parameters than their shallower counterparts. A DNN with a large set of parameters can easily overfit to smaller training datasets (Zhang *et al.*, 2016). Therefore, during model design it is typical to strike a balance between the number of model parameters and the size of the data available for training. Data for ad-hoc retrieval mainly consists of,

- Corpus of search queries

- Corpus of candidate documents

- Ground truth—in the form of either explicit human relevance judgments or implicit labels (*e.g.*, from clicks)—for query-document pairs

While both large scale corpora of search queries (Pass *et al.*, 2006; Craswell *et al.*, 2009) and documents (Callan *et al.*, 2009; Craswell *et al.*, 2003; Bailey *et al.*, 2003) are publicly available for IR research, the amount of relevance judgments that can be associated with them are often limited outside of large industrial research labs—mostly due to user privacy concerns. We note that we are interested in datasets where

the raw text of the query and the document is available. Therefore, this excludes large scale public labelled datasets for learning-to-rank (*e.g.*, (Liu *et al.*, 2007)) that don't contain the textual contents.

The proportion of labelled and unlabelled data that is available influences the *level of supervision* that can be employed for training these deep models. Most of the models we covered in §4 operate under the data regime where large corpus of documents or queries is available, but limited (or no) labelled data. Under such settings where no direct supervision or relevance judgments is provided, typically an *unsupervised* approach is employed (*e.g.*, (Salakhutdinov and Hinton, 2009)). The unlabelled document (or query) corpus is used to learn good text representations, and then these learnt representations are incorporated into an existing retrieval model or a query-document similarity metric. If small amounts of labelled data are available, then that can be leveraged to train a retrieval model with few parameters that in turn uses text representations that is pre-trained on larger unlabelled corpus. Examples of such *semi-supervised* training includes models such as (Pang *et al.*, 2016b; Pang *et al.*, 2016a; Guo *et al.*, 2016a). In contrast, *fully-supervised* models—*e.g.*, (Huang *et al.*, 2013; Severyn and Moschitti, 2015; Mitra *et al.*, 2017a; Nanni *et al.*, 2017; Cohen and Croft, 2016)—optimize directly for the target task by training on large number of labelled query-document pairs.

✎ DNNs for IR can learn text representations *in situ*, or use pre-trained embeddings. When pre-trained embeddings are employed, care should be taken to make sure the representations are suitable for the task.

It is also useful to distinguish between deep neural models that focus on ranking long documents, from those that rank short texts (*e.g.*, for the question-answering task, or for document ranking where the document

representation is based on a short text field like title). The challenges in short text ranking are somewhat distinct from those involved in the ad-hoc retrieval task (Cohen *et al.*, 2016). When computing similarity between pairs of short-texts, vocabulary mismatches are more likely than when the retrieved items contain long text descriptions (Metzler *et al.*, 2007). Neural models that perform matching in a latent space tend to be more robust towards the vocabulary mismatch problem compared to lexical term-based matching models. On the other hand, documents with long body texts may contain mixture of many topics and the query matches may be spread over the whole document. A neural document ranking model must effectively aggregate the relevant matches from different parts of a long document. In the rest of this section, we discuss several neural architectures and approaches to document ranking.

7.1 Document autoencoders

Salakhutdinov and Hinton (2009) proposed *Semantic Hashing*—one of the earliest deep neural models for ad-hoc retrieval. The model is a deep autoencoder trained under unsupervised setting on unlabelled document collection. The model considers each document as a bag-of-terms and uses one-hot vector representation for the terms—considering only top two thousand most popular terms in the corpus after removing stopwords. Salakhutdinov and Hinton (2009) first pre-train the model layer-by-layer, and then train it further end-to-end for additional tuning. After fine tuning the output of the model are thresholded to generate binary vector encoding of the documents. Given a search query, a corresponding hash is generated, and the relevant candidate documents quickly retrieved that match the same hash vector. A standard IR model can then be employed to rank between the selected documents.

Semantic hashing is an example of a document encoder based approach to IR. Variational autoencoders have also been explored (Chaidaroon and Fang, 2017) on similar lines. While vocabulary sizes of few thousand distinct terms may be too small for most practical IR tasks, a larger vocabulary or a different term representation strategy—such as the character trigraph based representation of Figure 6.3c—may be considered in practice. Another shortcoming of the autoencoder

architecture is that it minimizes the document reconstruction error which may not align well with the goal of the target IR task. A better alternative may be to train on query-document paired data where the choice of what constitutes as the minimal sufficient statistics of the document is influenced by what is important for determining relevance of the document to likely search queries. In line with this intuition, we next discuss the Siamese architecture based models.

7.2 Siamese networks

In recent years, several deep neural models based on the Siamese architecture have been explored especially for short text matching. The *Deep Semantic Similarity Model* (DSSM) (Huang *et al.*, 2013) is one such architecture that trains on query and document title pairs where both the pieces of texts are represented as bags-of-character-trigraphs. The DSSM architecture consists of two deep models—for the query and the document—with all fully-connected layers and cosine distance as the choice of similarity function in the middle. Huang *et al.* (2013) proposed to train the model on clickthrough data where each training sample consists of a query q, a positive document d^+ (a document that was clicked by a user on the SERP for that query), and a set of negative documents D^- randomly sampled with uniform probability from the full collection. The model is trained my minimizing the cross-entropy loss,

$$\mathcal{L}_{dssm}(q, d^+, D^-) = -\log\Big(\frac{e^{\gamma \cdot cos(\vec{q}, \vec{d^+})}}{\sum_{d \in D} e^{\gamma \cdot cos(\vec{q}, \vec{d})}}\Big) \qquad (7.1)$$

$$\text{where,} \quad D = \{d^+\} \cup D^- \qquad (7.2)$$

While, DSSM (Huang *et al.*, 2013) employs deep fully-connected architecture for the query and the document models, more sophisticated architectures involving convolutional layers (Shen *et al.*, 2014b; Gao *et al.*, 2014; Shen *et al.*, 2014a; Hu *et al.*, 2014), recurrent layers (Palangi *et al.*, 2015; Palangi *et al.*, 2014), and tree-structured networks (Tai *et al.*, 2015) have also been explored. The similarity function can also be parameterized and implemented as additional layers of the neural

network as in (Severyn and Moschitti, 2015). Most of these models have been evaluated on the short text matching task, but Mitra *et al.* (2017a) recently reported meaningful performances on the long document ranking task from models like DSSM (Huang *et al.*, 2013) and CDSSM (Shen *et al.*, 2014a) under telescoping evaluation. Mitra *et al.* (2017a) also show that sampling the negative documents uniformly from the collection is less effective to using documents that are closer to the query intent but judged as non-relevant by human annotators in similar evaluation settings.

Notions of similarity   It is important to emphasize that our earlier discussion in §3.2 on different notions of similarity between terms that can be learnt by shallow embedding models is also relevant in the context of these deeper architectures. In the case of Siamese networks, such as the convolutional-DSSM (CDSSM) (Shen *et al.*, 2014a), the notion of similarity being modelled depends on the choice of the paired data that the model is trained on. When the CDSSM is trained on query and document title pairs (Shen *et al.*, 2014a) then the notion of similarity is more *topical* in nature. Mitra and Craswell (2015) trained the same CDSSM architecture on query prefix-suffix pairs which, in contrast, captures a more *typical* notion of similarity, as shown in Table 7.1. In a related work, Mitra (2015) demonstrated that the CDSSM model when trained on session-query pairs is amenable to vector-based text analogies.

$$\vec{v}_{\text{things to do in london}} - \vec{v}_{\text{london}} + \vec{v}_{\text{new york}} \approx \vec{v}_{\text{new york tourist attractions}}$$
$$(7.3)$$

$$\vec{v}_{\text{university of washington}} - \vec{v}_{\text{seattle}} + \vec{v}_{\text{denver}} \approx \vec{v}_{\text{university of colorado}} \quad (7.4)$$

$$\vec{v}_{\text{new york}} + \vec{v}_{\text{newspaper}} \approx \vec{v}_{\text{new york times}} \quad (7.5)$$

By modelling different notions of similarity these deep neural models tend to be more suitable for other IR tasks, such as query auto-completion (Mitra and Craswell, 2015) or session-based personalization (Mitra, 2015).

Table 7.1: Comparing the nearest neighbours for "seattle" and "taylor swift" in the CDSSM embedding spaces when the model is trained on query-document pairs vs. query prefix-suffix pairs. The former resembles a topical notion of similarity between terms while the latter is more typical in the definition of inter-term similarities.

seattle		taylor swift	
Query-Document	Prefix-Suffix	Query-Document	Prefix-Suffix
weather seattle	chicago	taylor swift.com	lady gaga
seattle weather	san antonio	taylor swift lyrics	meghan trainor
seattle washington	denver	how old is taylor swift	megan trainor
ikea seattle	salt lake city	taylor swift twitter	nicki minaj
west seattle blog	seattle wa	taylor swift new song	anna kendrick

7.3 Interaction-based networks

Siamese networks represent both the query and the document using single embedding vectors. Alternatively, we can individually compare different parts of the query with different parts of the document, and then aggregate these partial evidences of relevance. Especially, when dealing with long documents—that may contain a mixture of many topics—such a strategy may be more effective than trying to represent the full document as a single low-dimensional vector. Typically, in these approaches a sliding window is moved over both the query and the document text and each instance of the window over the query is compared (or "interacts") against each instance of the window over the document text (see Figure 7.1). The terms within each window can be represented in different ways including, one-hot vectors, pre-trained embeddings, or embeddings that are updated during the model training. A neural model (typically convolutional) operates over the generated interaction matrix and aggregates the evidence across all the pairs of windows compared.
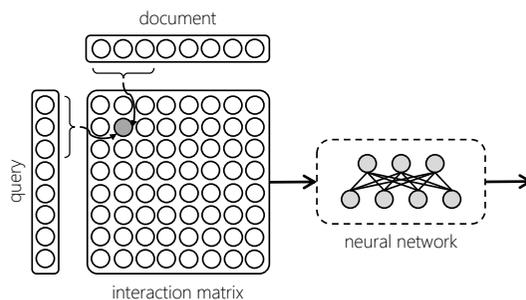
Figure 7.1: Schematic view of an interaction matrix generated by comparing windows of text from the query and the document. A deep neural network—such as a CNN—operates over the interaction matrix to find patterns of matches that suggest relevance of the document to the query.

> Long documents may contain a mixture of many topics. So, comparing the query representation individually to windows of text from the document may be more effective than learning a single vector representation for the document.

The interaction matrix based approach have been explored both for short text matching (Lu and Li, 2013; Hu *et al.*, 2014; Yin *et al.*, 2015; Pang *et al.*, 2016b; Yang *et al.*, 2016; Wan *et al.*, 2015), as well as for ranking long documents (Mitra *et al.*, 2017a; Pang *et al.*, 2016a; Hui *et al.*, 2017; Hui *et al.*, 2018).

7.4 Lexical and semantic matching

Much of the explorations in neural IR models have focused on learning good representations of text. However, these representation learning models tend to perform poorly when dealing with rare terms and search intents. In §2.2, we highlighted the importance of modelling rare terms

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

(a) Lexical model

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

(b) Semantic model

Figure 7.2: Analysis of term importance for estimating the relevance of a passage to the query "United States President" by a lexical and a semantic deep neural network model. The lexical model only considers the matches of the query terms in the document but gives more emphasis to earlier occurrences. The semantic model is able to extract evidence of relevance from related terms such as "Obama" and "federal".

in IR. Based on similar motivations, Guo *et al.* (2016a) and Mitra *et al.* (2017a) have recently emphasized the importance of modelling lexical matches using deep neural networks. Mitra *et al.* (2017a) argue that Web search is a "tale of two queries". For the query "pekarovic land company", it is easier to estimate relevance based on patterns of exact matches of the rare term "pekarovic". On the other hand, a neural model focused on matching in the latent space is unlikely to have a good representation for this rare term. In contrast, for the query "what channel are the seahawks on today", the target document likely contains "ESPN" or "Sky Sports"—not the term "channel". A representation learning neural model can associate occurrences of "ESPN" in the document as positive evidence towards the document being relevant to the query. Figure 7.2 highlights the difference between the terms that influence the estimation of relevance of the same query-passage pair by a lexical matching and a semantic matching model. A good neural IR
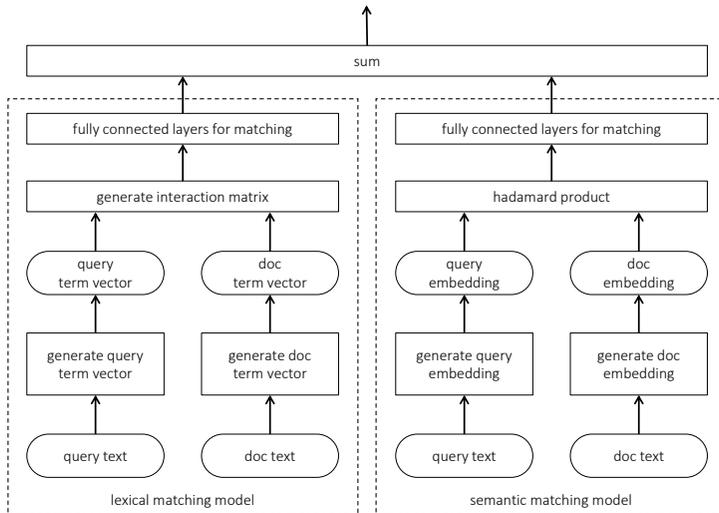
Figure 7.3: In the Duet architecture (Mitra *et al.*, 2017a), the two sub-networks are jointly trained and the final output is a linear combination of the outputs of the lexical and the semantic matching sub-networks. The lexical matching sub-network (left) uses a convolutional model that operates over a binary interaction matrix.[1] The semantic matching sub-network (right) learns representations of query and document text for effective matching in the latent space. Cross-entropy loss is used to train the network similar to other models in §7.2.

model should incorporate both lexical and semantic matching signals (Mitra *et al.*, 2017a).

Guo *et al.* (2016a) proposed to use histogram-based features in their DNN model to capture lexical notion of relevance. Mitra *et al.* (2017a) leverage large scale labelled data from Bing to train a *Duet* architecture[2] (Figure 7.3) that learns to identify good patterns of both lexical and semantic matches jointly. Neural models that focus on lexical

---

[1]It is important to emphasize, that while Mitra *et al.* (2017a) and others have used interaction-based representation for modelling lexical matches, the two ideas are distinct. Some interaction-matrix based representations compare texts using their pre-trained embeddings (Hu *et al.*, 2014; Yin *et al.*, 2015). Similarly, lexical matching can be modelled without employing an interaction matrix based representation (Guo *et al.*, 2016a).

[2]A CNTK implementation of the duet architecture is available at https://github.com/bmitra-msft/NDRM/blob/master/notebooks/Duet.ipynb
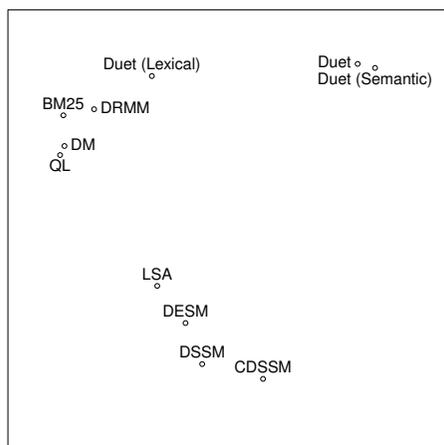
Figure 7.4: A demonstration that IR models that focus on lexical matching tend to perform well on queries that are distinct from queries on which semantic matching models achieve good relevance. Each model is represented by a vector of NDCG scores achieved on a set of test queries. For visualization, t-SNE (Maaten and Hinton, 2008) is used to plot the points in a two-dimensional space. Lexical matching models (BM25, QL, DM, DRMM, and Duet-Lexical) are seen to form a cluster—as well as the models that focus on representation learning.

matching typically have fewer parameters and can be trained under small data regimes—unlike their counterparts that focus on learning representations of text.

A query level analysis indicates that both traditional non-neural IR approaches and more recent neural methods tend to perform well on different segments of queries depending on whether they focus on lexical or semantic matching. Figure 7.4 plots a few of these models based on their per-query NDCG values on a set of test queries.

✎ IR models that use exact matching tend to perform well on different queries than models that employ inexact matching. Models that combine both performs best.

7.5 Matching with multiple document fields

Commercial Web search engines, such as Google and Bing, consider more than just the document content for matching. Anchor texts corresponding to incoming hyperlinks and the query text for which the document may have been previously viewed are among the various sources that may provide additional description of the document. Incorporating these different sources of document description is useful to determine the relevance of the document to a search query (Robertson *et al.*, 2004). It is, therefore, important to study neural models that can effectively leverage multiple document fields for the ranking task.

Zamani *et al.* (2018) recently studied some of the principles for designing neural architectures for ranking documents with multiple fields. They find that learning separate latent spaces for matching the query against the different document fields is more effective than using a shared latent space for all the fields. This may be because the different document fields may contain information relevant to different aspects of the query intent. For example, for the query "roar music video" shown in Table 7.2, the URL sub-model of the deep neural network may learn a latent space that considers the implicit domain intent in the query, which may be different than the query representation that the title

Table 7.2: Different document fields may be useful for matching different aspects of the query intent. For example, for the query "roar music video" the document title may be useful for matching the name of the song, whereas the URL may be more informative on whether the document is likely to satisfy the video intent. A neural representation learning model may emphasize on the video intent in the query embedding that it learns to match against the URL field and pay more attention to matching the name of the song against the title field. By combining the two signals the model can better discriminate the relevant document from the rest.

	Text fields		Field matches		
	URL	Title	URL	Title	Combined
Doc #1	youtube.com/watch?v=CevxZvSJLk8	Katy Perry - Roar	✓	✓	✓
Doc #2	youtube.com/watch?v=nfWlot6h_JM	Taylor Swift - Shake It Off	✓	✗	✗
Doc #3	wikipedia.org/wiki/Roar_(song)	Roar (song)	✗	✓	✗
Doc #4	wikipedia.org/wiki/Shake_It_Off	Shake It Off	✗	✗	✗

sub-model may find appropriate for matching. Similar to the findings
by Robertson *et al.* (2004), they also observe that learning a document
representation based on all the available fields is superior to combining
the relevance scores computed on each individual field. Finally, a field-
level dropout is proposed for regularizing against over-dependency
on any individual field—*e.g.* the clicked queries field— during model
training. The architecture proposed by Zamani *et al.* (2018) is shown
in Figure 7.5.
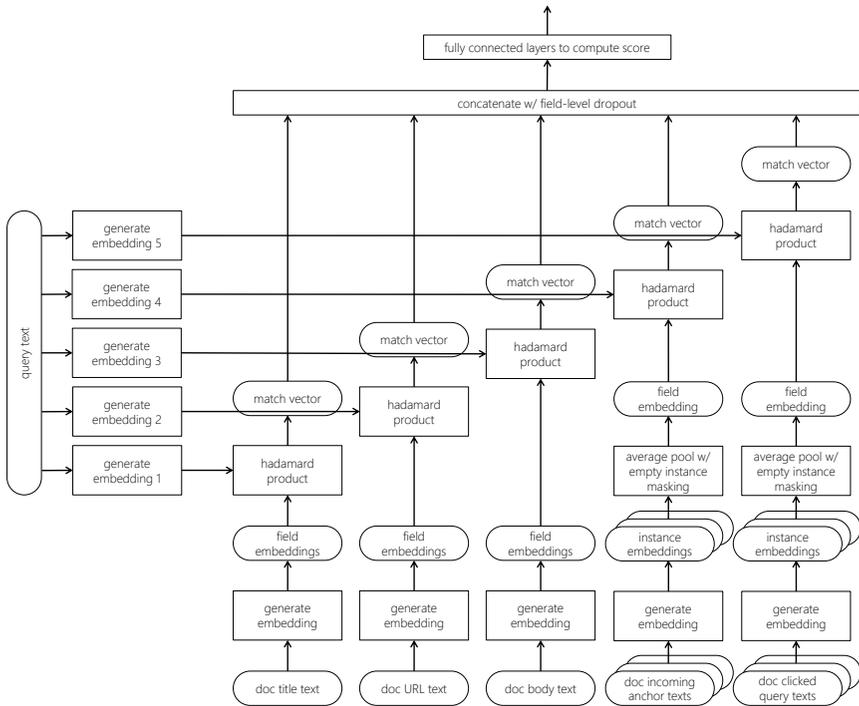
Figure 7.5: A neural architecture for ranking documents with multiple document
fields. A new latent space is learnt—corresponding to each field—and the query
and the document are compared in each of these latent spaces to generate a match
vector. The relevance of the document to the query is estimated by aggregating the
matching signals across the different latent spaces.

8

Conclusion

We present a tutorial on neural methods for information retrieval. For machine learning researchers who may be less familiar with IR tasks, we introduced the fundamentals of traditional IR models and metrics. For IR researchers, we summarized key concepts related to representation learning with (shallow or deep) neural networks. Finally, we presented some of the recent neural methods for document ranking and question-answer matching.

We have focused on retrieval of long and short text. In the case of long text, the model must deal with variable length documents, where the relevant sections of a document may be surrounded by irrelevant text. For both long and short text, but particularly for short, IR models should also deal with the query-document vocabulary mismatch problem, by learning how patterns of query terms and (different) document terms can indicate relevance. Models should also consider lexical matches when the query contains rare terms—such as a person's name or a product model number—not seen during training, and to avoid retrieving semantically related but irrelevant results.

An ideal IR model would be able to infer the meaning of a query from context. Given a query about the Prime Minister of UK, for

93

example, it may be obvious from context whether it refers to John Major or Teresa May—perhaps due to the time period of the corpus, or it may need to be disambiguated based on other context such as the other query terms or the user's short or long-term history. If the model learns a representation that encodes this context, perhaps making Prime Minister close to Teresa May in a latent space, it is like a *library*. To scale to a large corpus, this memorization would need to cover a massive number of connections between entities and contexts, which could potentially be limited by model capacity. Memorization could also cause update problems, for example if there is a new Prime Minister but the model and most documents still refer to the old one. To avoid these problems, another design could avoid memorizing connections in the corpus, and instead perform some per-query process that reads the corpus and perhaps even reasons about the content, like a *librarian*.

> Should the ideal IR model behave like a *library* that knows about everything in the Universe, or like a *librarian* who can effectively retrieve without memorizing the corpus?

Many of the breakthroughs in deep learning have been motivated by the needs of specific application areas. Convolutional neural networks, for example, are particularly popular with the vision community, whereas recurrent architectures find more applications in speech recognition and NLP. It is likely that the specific nature of IR tasks and data will inform our choice of neural architectures and drive us towards new designs. Future IR explorations may also be motivated by developments in related areas, such as NLP. Neural architectures that have been evaluated on non-IR tasks (Zhao *et al.*, 2015; Kalchbrenner *et al.*, 2014; Denil *et al.*, 2014; Kim, 2014; Collobert *et al.*, 2011b) can be investigated in the retrieval context. New methods for training neural IR models—*e.g.*, using reinforcement (Nogueira and Cho, 2017; Buck *et al.*, 2017; Rosset

et al., 2018) or adversarial learning (Wang *et al.*, 2017; Cohen *et al.*, 2018)—may also emerge as important directions for future explorations.

However, given the pace at which the area of deep learning is growing, in terms of the number of new architectures and training regimes, we should be wary of the combinatorial explosion of trying every model on every IR task. We should not disproportionately focus on maximizing quantitative improvements and in the process, neglect theoretical understanding and qualitative insights. It would be a bad outcome for the field if these explorations do not grow our understanding of the fundamental principles of machine learning and information retrieval. Neural models should not be the hammer that we try on every IR task, or we may risk reducing every IR task to a nail.[1] Rather, these new models should also be the lens through which researchers gain new insights into the underlying principles of IR tasks. This may imply that sometimes we prefer neural models that, if not interpretable, then at least are amenable to analysis and interrogation. We may elicit more insights from simpler models while more sophisticated models may achieve state-of-the-art performances. As a community, we may need to focus on both to achieve results that are both impactful as well as insightful.

> ✎ The hope of neural IR is that the unique challenges of IR tasks will motivate new breakthroughs in deep learning, as well as these new models will grow our understanding of core IR tasks.

The focus of this tutorial has been on ad-hoc retrieval and to a lesser extent on question-answering. However, neural approaches have shown interesting applications to other existing retrieval scenarios, including query auto-completion (Mitra and Craswell, 2015), query recommendation (Sordoni *et al.*, 2015a), session modelling (Mitra, 2015),

---

[1]https://en.wikipedia.org/wiki/Law_of_the_instrument

modelling diversity (Xia *et al.*, 2016), modelling user click behaviours (Borisov *et al.*, 2016b; Borisov *et al.*, 2016a), proactive recommendations (Luukkonen *et al.*, 2016; Van Gysel *et al.*, 2017b), entity ranking (Van Gysel *et al.*, 2016a; Van Gysel *et al.*, 2016b), multi-modal retrieval (Ma *et al.*, 2015), knowledge-based IR (Nguyen *et al.*, 2016a), conversational agents (Yan *et al.*, 2016; Zhou *et al.*, 2016), and even optimizing for multiple IR tasks (Liu *et al.*, n.d.). In addition, recent trends suggest that advances in deep neural networks methods are also fuelling emerging IR scenarios such as conversational IR (Yan *et al.*, 2016; Zhou *et al.*, 2016) and multi-modal retrieval (Ma *et al.*, 2015). Neural methods may have an even bigger impact on some of these other IR tasks, perhaps enabling applications that were previously impossible.

IR also has a role in the context of the ambitions of the machine learning community. Retrieval is key to many one-shot learning approaches (Koch, 2015; Vinyals *et al.*, 2016). Ghazvininejad *et al.* (2017) proposed to "search" external information sources in the process of solving complex tasks using neural networks. The idea of learning local representations proposed by Diaz *et al.* (2016) may be applicable to non-IR tasks. While we look at applying neural methods to IR, we should also look for opportunities to leverage IR techniques as part of—or in combination with—neural and other machine learning models.

Finally, we must also renew our focus on the fundamentals, including benchmarking and reproducibility. An important prerequisite to enable the "neural IR train" to steam forward is to build shared public resources—*e.g.*, large scale datasets for training and evaluation, and repository of shared model implementations—and to ensure that appropriate bindings exist (*e.g.*, (Van Gysel *et al.*, 2017a; Mitra *et al.*, 2017b)) between popular IR frameworks and popular toolkits from the neural network community. At the time of writing, the IR community does not yet have large shared label sets for training and testing deep models. This problem could be addressed by a data generation and sharing initiative of sufficient scale, comparable to the computer vision community's ImageNet database (Russakovsky *et al.*, 2015).

The emergence of new IR tasks also demands rethinking many of our existing metrics. The metrics that may be appropriate for evaluating document ranking systems may be inadequate when the system

generates textual answers in response to information seeking questions. In the latter scenario, the metric should distinguish between whether the response differs from the ground truth in the information content or in phrasing of the answer (Mitra *et al.*, 2016b; Liu *et al.*, 2016; Galley *et al.*, 2015). As multi-turn interactions with retrieval systems become more common, the definition of task success will also need to evolve accordingly. Neural IR should not only focus on novel techniques, but also encompass all these other aspects.

Acknowledgements

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, O. Viegas Fernanda and-Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2016. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". *arXiv preprint arXiv:1603.04467.*

Abdul-Jaleel, N., J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, M. D. Smucker, and C. Wade. 2004. "UMass at TREC 2004: Novelty and HARD".

Agichtein, E., D. Carmel, D. Pelleg, Y. Pinter, and D. Harman. 2015. "Overview of the TREC 2015 LiveQA Track." In: *TREC.*

Ai, Q., L. Yang, J. Guo, and W. B. Croft. 2016a. "Analysis of the paragraph vector model for information retrieval". In: *Proc. ICTIR.* ACM. 133–142.

Ai, Q., L. Yang, J. Guo, and W. B. Croft. 2016b. "Improving language estimation with the paragraph vector model for ad-hoc retrieval". In: *Proc. SIGIR.* ACM. 869–872.

Arora, S., Y. Li, Y. Liang, T. Ma, and A. Risteski. 2015. "Rand-walk: A latent variable model approach to word embeddings". *arXiv preprint arXiv:1502.03520.*

Baeza-Yates, R. and B. Ribeiro-Neto. 1999. *Modern information retrieval.* Vol. 463. ACM press New York.

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

Bahuleyan, H., L. Mou, O. Vechtomova, and P. Poupart. 2017. "Variational Attention for Sequence-to-Sequence Models". *arXiv preprint arXiv:1712.08207.*

Bailey, P., N. Craswell, and D. Hawking. 2003. "Engineering a multi-purpose test collection for web retrieval experiments". *Information Processing & Management.* 39(6): 853–871.

Baldi, P. and Y. Chauvin. 1993. "Neural networks for fingerprint recognition". *Neural Computation.* 5(3): 402–418.

Baroni, M., G. Dinu, and G. Kruszewski. 2014. "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors". In: *Proc. ACL.* Vol. 1. 238–247.

Baroni, M. and A. Lenci. 2010. "Distributional memory: A general framework for corpus-based semantics". *Computational Linguistics.* 36(4): 673–721.

Barthes, R. 1977. *Elements of semiology.* Macmillan.

Benetka, J. R., K. Balog, and K. Nørvåg. 2017. "Anticipating Information Needs Based on Check-in Activity". In: *WSDM.* ACM. 41–50.

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

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin. 2003. "A neural probabilistic language model". *Journal of machine learning research.* 3(Feb): 1137–1155.

Bengio, Y., P. Lamblin, D. Popovici, H. Larochelle, *et al.* 2007. "Greedy layer-wise training of deep networks". *Proc. NIPS.* 19: 153.

Bengio, Y. and J.-S. Senécal. 2008. "Adaptive importance sampling to accelerate training of a neural probabilistic language model". *IEEE Transactions on Neural Networks.* 19(4): 713–722.

Bengio, Y., J.-S. Senécal, *et al.* 2003. "Quick Training of Probabilistic Neural Nets by Importance Sampling." In: *AISTATS.*

Berger, A. and J. Lafferty. 1999. "Information retrieval as statistical translation". In: *Proc. SIGIR.* ACM. 222–229.

Bhatia, K., P. Jain, and M. Varma. 2016. "The extreme classification repository: multi-label datasets & code". Accessed June 7, 2017. URL: http://manikvarma.org/downloads/XC/XMLRepository.html.

Bishop, C. M. 2006. *Pattern recognition and machine learning.* Springer-Verlag New York.

Blei, D. M., A. Y. Ng, and M. I. Jordan. 2003. "Latent dirichlet allocation". *the Journal of machine Learning research.* 3: 993–1022.

Bordes, A., N. Usunier, S. Chopra, and J. Weston. 2015. "Large-scale simple question answering with memory networks". *arXiv preprint arXiv:1506.02075.*

Borisov, A., I. Markov, M. de Rijke, and P. Serdyukov. 2016a. "A context-aware time model for web search". In: *Proc. SIGIR.* ACM. 205–214.

Borisov, A., I. Markov, M. de Rijke, and P. Serdyukov. 2016b. "A neural click model for web search". In: *Proc. WWW.* 531–541.

Bowman, S. R., J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts. 2016. "A fast unified model for parsing and sentence understanding". *arXiv preprint arXiv:1603.06021.*

Brébisson, A. de and P. Vincent. 2015. "An exploration of softmax alternatives belonging to the spherical loss family". *arXiv preprint arXiv:1511.05042.*

Brébisson, A. de and P. Vincent. 2016. "The Z-loss: a shift and scale invariant classification loss belonging to the Spherical Family". *arXiv preprint arXiv:1604.08859.*

Brill, E. 2003. "Processing natural language without natural language processing". In: *International Conference on Intelligent Text Processing and Computational Linguistics.* Springer. 360–369.

Bromley, J., J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. 1993. "Signature Verification Using A "Siamese" Time Delay Neural Network". *IJPRAI.* 7(4): 669–688.

Brown, P. F., J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. "A statistical approach to machine translation". *Computational linguistics.* 16(2): 79–85.

Brown, P. F., P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. 1992. "Class-based n-gram models of natural language". *Computational linguistics.* 18(4): 467–479.

Brown, P. F., V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. 1993. "The mathematics of statistical machine translation: Parameter estimation". *Computational linguistics.* 19(2): 263–311.

Buck, C., J. Bulian, M. Ciaramita, A. Gesmundo, N. Houlsby, W. Gajewski, and W. Wang. 2017. "Ask the right questions: Active question reformulation with reinforcement learning". *arXiv preprint arXiv:1705.07830.*

Bullinaria, J. A. and J. P. Levy. 2007. "Extracting semantic representations from word co-occurrence statistics: A computational study". *Behavior research methods.* 39(3): 510–526.

Bullinaria, J. A. and J. P. Levy. 2012. "Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD". *Behavior research methods.* 44(3): 890–907.

Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. 2005. "Learning to rank using gradient descent". In: *Proc. ICML.* ACM. 89–96.

Burges, C. J. 2010. "From ranknet to lambdarank to lambdamart: An overview". *Learning.* 11(23-581): 81.

Burges, C. J., R. Ragno, and Q. V. Le. 2006. "Learning to rank with nonsmooth cost functions". In: *NIPS.* Vol. 6. 193–200.

Callan, J., M. Hoy, C. Yoo, and L. Zhao. 2009. "Clueweb09 data set".

Cao, Z., T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. 2007. "Learning to rank: from pairwise approach to listwise approach". In: *Proc. ICML.* ACM. 129–136.

Chaidaroon, S. and Y. Fang. 2017. "Variational Deep Semantic Hashing for Text Documents". In: *Proc. SIGIR.*

Chandler, D. 1994. "Semiotics for beginners".

Chen, T., M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. 2015a. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". *arXiv preprint arXiv:1512.01274.*

Chen, W., T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li. 2009. "Ranking measures and loss functions in learning to rank". In: *Advances in Neural Information Processing Systems.* 315–323.

Chen, W., D. Grangier, and M. Auli. 2015b. "Strategies for training large vocabulary neural language models". *arXiv preprint arXiv:1512.04906.*

Cho, K., B. Van Merriënboer, D. Bahdanau, and Y. Bengio. 2014. "On the properties of neural machine translation: Encoder-decoder approaches". *arXiv preprint arXiv:1409.1259.*

Chopra, S., R. Hadsell, and Y. LeCun. 2005. "Learning a similarity metric discriminatively, with application to face verification". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 1. IEEE. 539–546.

Chorowski, J. K., D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. 2015. "Attention-based models for speech recognition". In: *Proc. NIPS.* 577–585.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio. 2014. "Empirical evaluation of gated recurrent neural networks on sequence modeling". *arXiv preprint arXiv:1412.3555.*

Cisse, M., M. Varma, and S. Bengio. 2016. "Extreme Classification 2016: The NIPS Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces". Accessed June 7, 2017. URL: http://manikvarma.org/events/XC16/.

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

Cohen, D., Q. Ai, and W. B. Croft. 2016. "Adaptability of neural networks on varying granularity IR tasks". *arXiv preprint arXiv:1606.07565.*

Cohen, D. and W. B. Croft. 2016. "End to End Long Short Term Memory Networks for Non-Factoid Question Answering". In: *Proc. ICTIR*. ACM. 143–146.

Cohen, D., B. Mitra, K. Hofmann, and B. Croft. 2018. "Cross Domain Regularization for Neural Ranking Models using Adversarial Learning". In: *Proc. SIGIR*. ACM.

Collobert, R., K. Kavukcuoglu, and C. Farabet. 2011a. "Torch7: A matlab-like environment for machine learning". In: *BigLearn, NIPS Workshop*. No. EPFL-CONF-192376.

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

Cossock, D. and T. Zhang. 2006. "Subset ranking using regression". In: *COLT*. Vol. 6. Springer. 605–619.

Craswell, N. 2009. "Mean reciprocal rank". In: *Encyclopedia of Database Systems*. Springer. 1703–1703.

Craswell, N. 2017. "Neural Models for Full Text Search". In: *Proc. WSDM*. ACM. 251–251.

Craswell, N., W. B. Croft, J. Guo, B. Mitra, and M. de Rijke. 2016a. "Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval".

Craswell, N., W. B. Croft, J. Guo, B. Mitra, and M. de Rijke. 2016b. "Report on the SIGIR 2016 Workshop on Neural Information Retrieval (Neu-IR)". *ACM Sigir forum*. 50(2): 96–103.

Craswell, N., W. B. Croft, M. de Rijke, J. Guo, and B. Mitra. 2017. "Neu-IR'17: Neural Information Retrieval". In: *Proc. SIGIR*. ACM.

Craswell, N., W. B. Croft, M. de Rijke, J. Guo, and B. Mitra. 2018. "Report on the Second SIGIR Workshop on Neural Information Retrieval (Neu-IR'17)". In: *ACM SIGIR Forum*. Vol. 51. No. 3. ACM. 152–158.

Craswell, N., D. Hawking, R. Wilkinson, and M. Wu. 2003. "Overview of the TREC 2002 Web track". In: *TREC*. Vol. 3. 12th.

Craswell, N., R. Jones, G. Dupret, and E. Viegas. 2009. *Proceedings of the 2009 workshop on Web Search Click Data*. ACM.

Croft, W. B., D. Metzler, and T. Strohman. 2010. *Search engines: Information retrieval in practice*. Vol. 283. Addison-Wesley Reading.

De Saussure, F. 1916. "Cours de linguistique générale, publié par Ch". *Bally et A. Sechehaye avec la collaboration de A. Riedlinger. Paris: Payot.*

Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. "Indexing by latent semantic analysis". *JASIS.* 41(6): 391–407.

Dehghani, M., A. Severyn, S. Rothe, and J. Kamps. 2017a. "Learning to Learn from Weak Supervision by Full Supervision". *arXiv preprint arXiv:1711.11383.*

Dehghani, M., H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. 2017b. "Neural ranking models with weak supervision". In: *Proc. SIGIR.* ACM. 65–74.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the royal statistical society. Series B (methodological)*: 1–38.

Deng, L., D. Yu, *et al.* 2014. "Deep learning: methods and applications". *Foundations and Trends® in Signal Processing.* 7(3–4): 197–387.

Denil, M., A. Demiraj, N. Kalchbrenner, P. Blunsom, and N. de Freitas. 2014. "Modelling, visualising and summarising documents with a single convolutional neural network". *arXiv preprint arXiv:1406.3830.*

Diaz, F., B. Mitra, and N. Craswell. 2016. "Query Expansion with Locally-Trained Word Embeddings". In: *Proc. ACL.*

Diaz, F., R. White, G. Buscher, and D. Liebling. 2013. "Robust models of mouse movement on dynamic web search results pages". In: *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management.* ACM. 1451–1460.

Donmez, P., K. M. Svore, and C. J. Burges. 2009. "On the local optimality of LambdaRank". In: *Proc. SIGIR.* ACM. 460–467.

Duchi, J., E. Hazan, and Y. Singer. 2011. "Adaptive subgradient methods for online learning and stochastic optimization". *Journal of Machine Learning Research.* 12(Jul): 2121–2159.

Dyer, C. 2014. "Notes on Noise Contrastive Estimation and Negative Sampling". *arXiv preprint arXiv:1410.8251.*

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

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

Firth, J. R. 1957. "A synopsis of linguistic theory, 1930–1955". In: *Studies in Linguistic Analysis.* Blackwell. 1–32.

Freund, Y., R. Iyer, R. E. Schapire, and Y. Singer. 2003. "An efficient boosting algorithm for combining preferences". *Journal of machine learning research.* 4(Nov): 933–969.

Fuhr, N. 1989. "Optimum polynomial retrieval functions based on the probability ranking principle". *ACM Transactions on Information Systems (TOIS).* 7(3): 183–204.

Galley, M., C. Brockett, A. Sordoni, Y. Ji, M. Auli, C. Quirk, M. Mitchell, J. Gao, and B. Dolan. 2015. "deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets". *arXiv preprint arXiv:1506.06863.*

Ganguly, D., D. Roy, M. Mitra, and G. J. Jones. 2015. "Word Embedding based Generalized Language Model for Information Retrieval". In: *Proc. SIGIR.* ACM. 795–798.

Gao, J., P. Pantel, M. Gamon, X. He, L. Deng, and Y. Shen. 2014. "Modeling interestingness with deep neural networks". In: *Proc. EMNLP.*

Ghazvininejad, M., C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-t. Yih, and M. Galley. 2017. "A Knowledge-Grounded Neural Conversation Model". *arXiv preprint arXiv:1702.01932.*

Goldberg, Y. and O. Levy. 2014. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". *arXiv preprint arXiv:1402.3722.*

Goller, C. and A. Kuchler. 1996. "Learning task-dependent distributed representations by backpropagation through structure". In: *Neural Networks, 1996., IEEE International Conference on.* Vol. 1. IEEE. 347–352.

Golub, G. H. and C. Reinsch. 1970. "Singular value decomposition and least squares solutions". *Numerische mathematik.* 14(5): 403–420.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep learning.* MIT Press.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. "Generative adversarial nets". In: *Proc. NIPS.* 2672–2680.

Goodman, J. 2001. "Classes for fast maximum entropy training". In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP'01). 2001 IEEE International Conference on.* Vol. 1. IEEE. 561–564.

Granka, L. A., T. Joachims, and G. Gay. 2004. "Eye-tracking analysis of user behavior in WWW search". In: *Proc. SIGIR.* ACM. 478–479.

Grave, E., A. Joulin, M. Cissé, D. Grangier, and H. Jégou. 2016. "Efficient softmax approximation for GPUs". *arXiv preprint arXiv:1609.04309.*

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

Graves, A., M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. 2009. "A novel connectionist system for unconstrained handwriting recognition". *IEEE transactions on pattern analysis and machine intelligence.* 31(5): 855–868.

Grbovic, M., N. Djuric, V. Radosavljevic, and N. Bhamidipati. 2015a. "Search Retargeting using Directed Query Embeddings". In: *Proc. WWW.* International World Wide Web Conferences Steering Committee. 37–38.

Grbovic, M., N. Djuric, V. Radosavljevic, F. Silvestri, and N. Bhamidipati. 2015b. "Context-and Content-aware Embeddings for Query Rewriting in Sponsored Search". In: *Proc. SIGIR.* ACM. 383–392.

Gregor, K., I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. 2015. "DRAW: A recurrent neural network for image generation". *arXiv preprint arXiv:1502.04623.*

Guan, Z. and E. Cutrell. 2007. "An eye tracking study of the effect of target rank on web search". In: *Proc. SIGCHI.* ACM. 417–420.

Guo, J., Y. Fan, Q. Ai, and W. B. Croft. 2016a. "A Deep Relevance Matching Model for Ad-hoc Retrieval". In: *Proc. CIKM.* ACM. 55–64.

Guo, J., Y. Fan, Q. Ai, and W. B. Croft. 2016b. "Semantic Matching by Non-Linear Word Transportation for Information Retrieval". In: *Proc. CIKM.* ACM. 701–710.

Gutmann, M. and A. Hyvärinen. 2010. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In: *AISTATS*. Vol. 1. No. 2. 6.

Hadsell, R., S. Chopra, and Y. LeCun. 2006. "Dimensionality reduction by learning an invariant mapping". In: *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE. 1735–1742.

Hahnloser, R. H., R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. 2000. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". *Nature*. 405(6789): 947–951.

Halevy, A., P. Norvig, and F. Pereira. 2009. "The unreasonable effectiveness of data". *IEEE Intelligent Systems*. 24(2): 8–12.

Harris, R. 2001. *Saussure and his Interpreters*. Edinburgh University Press.

Harris, Z. S. 1954. "Distributional structure". *Word*. 10(2-3): 146–162.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. "The elements of statistical learning. 2001".

He, K., X. Zhang, S. Ren, and J. Sun. 2016. "Deep residual learning for image recognition". In: *Proc. CVPR*. 770–778.

Hecht-Nielsen, R. *et al.* 1988. "Theory of the backpropagation neural network." *Neural Networks*. 1(Supplement-1): 445–448.

Herbrich, R., T. Graepel, and K. Obermayer. 2000. "Large margin rank boundaries for ordinal regression".

Hermann, K. M., T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. "Teaching machines to read and comprehend". In: *Proc. NIPS*. 1693–1701.

Hiemstra, D. 2001. *Using language models for information retrieval*. Taaluitgeverij Neslia Paniculata.

Hill, F., A. Bordes, S. Chopra, and J. Weston. 2015. "The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations". *arXiv preprint arXiv:1511.02301*.

Hinton, G. E. 1984. "Distributed representations".

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

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

Hofmann, K., B. Mitra, F. Radlinski, and M. Shokouhi. 2014. "An Eye-tracking Study of User Interactions with Query Auto Completion". In: *Proc. CIKM.* ACM. 549–558.

Hofmann, T. 1999. "Probabilistic latent semantic indexing". In: *Proc. SIGIR.* ACM. 50–57.

Hornik, K., M. Stinchcombe, and H. White. 1989. "Multilayer feedforward networks are universal approximators". *Neural networks.* 2(5): 359–366.

Hu, B., Z. Lu, H. Li, and Q. Chen. 2014. "Convolutional neural network architectures for matching natural language sentences". In: *Proc. NIPS.* 2042–2050.

Huang, G., C. Guo, M. J. Kusner, Y. Sun, F. Sha, and K. Q. Weinberger. 2016. "Supervised Word Mover's Distance". In: *Proc. NIPS.* 4862–4870.

Huang, P.-S., X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. "Learning deep structured semantic models for web search using clickthrough data". In: *Proc. CIKM.* ACM. 2333–2338.

Hui, K., A. Yates, K. Berberich, and G. de Melo. 2017. "PACRR: A Position-Aware Neural IR Model for Relevance Matching". In: *Proc. EMNLP.* 1049–1058.

Hui, K., A. Yates, K. Berberich, and G. de Melo. 2018. "Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval". In: *Proceedings of the 11th ACM International Conference on Web Search and Data Mining. WSDM.* Vol. 18. 2.

Jain, H., Y. Prabhu, and M. Varma. 2016. "Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications". In: *Proc. SIGKDD.* ACM. 935–944.

Jarrett, K., K. Kavukcuoglu, Y. LeCun, *et al.* 2009. "What is the best multi-stage architecture for object recognition?" In: *Computer Vision, 2009 IEEE 12th International Conference on.* IEEE. 2146–2153.

Järvelin, K. and J. Kekäläinen. 2002. "Cumulated gain-based evaluation of IR techniques". *ACM Transactions on Information Systems (TOIS).* 20(4): 422–446.

Jean, S., K. Cho, R. Memisevic, and Y. Bengio. 2014. "On Using Very Large Target Vocabulary for Neural Machine Translation". *arXiv preprint arXiv:1412.2007.*

Jelinek, F. and R. Mercer. 1980. "Interpolated estimation of Markov source parameters from sparse data". In: *Proc. Workshop on Pattern Recognition in Practice, 1980.*

Ji, S., S. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey. 2015. "Blackout: Speeding up recurrent neural network language models with very large vocabularies". *arXiv preprint arXiv:1511.06909.*

Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. 2014. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia.* ACM. 675–678.

Joachims, T., L. Granka, B. Pan, H. Hembrooke, and G. Gay. 2005. "Accurately interpreting clickthrough data as implicit feedback". In: *Proc. SIGIR.* Acm. 154–161.

Joachims, T., L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. 2007. "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search". *ACM Transactions on Information Systems (TOIS).* 25(2): 7.

Joachims, T., A. Swaminathan, and T. Schnabel. 2017. "Unbiased learning-to-rank with biased feedback". In: *Proc. WSDM.* ACM. 781–789.

Jozefowicz, R., O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. 2016. "Exploring the limits of language modeling". *arXiv preprint arXiv:1602.02410.*

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

Kenter, T., A. Borisov, and M. de Rijke. 2016. "Siamese cbow: Optimizing word embeddings for sentence representations". *arXiv preprint arXiv:1606.04640*.

Kenter, T., A. Borisov, C. Van Gysel, M. Dehghani, M. de Rijke, and B. Mitra. 2017. "Neural Networks for Information Retrieval (NN4IR)". In: *Proc. SIGIR*. ACM.

Kenter, T., A. Borisov, C. Van Gysel, M. Dehghani, M. de Rijke, and B. Mitra. 2018a. "Neural networks for information retrieval". In: *Proc. WSDM*. ACM. 779–780.

Kenter, T., A. Borisov, C. Van Gysel, M. Dehghani, M. de Rijke, and B. Mitra. 2018b. "Neural networks for information retrieval". In: *Proc. ECIR*.

Kenter, T. and M. de Rijke. 2015. "Short Text Similarity with Word Embeddings". In: *Proc. CIKM*. Vol. 15. 115.

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

Kim, Y., Y. Jernite, D. Sontag, and A. M. Rush. 2015. "Character-aware neural language models". *arXiv preprint arXiv:1508.06615*.

Kingma, D. P. and M. Welling. 2014. "Auto-encoding variational bayes". In: *Proc. ICLR*.

Kiros, R., R. Zemel, and R. R. Salakhutdinov. 2014. "A multiplicative model for learning distributed text-based attribute representations". In: *Proc. NIPS*. 2348–2356.

Koch, G. 2015. "Siamese neural networks for one-shot image recognition". *PhD thesis*. University of Toronto.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. "Imagenet classification with deep convolutional neural networks". In: *Proc. NIPS*. 1097–1105.

Kusner, M., Y. Sun, N. Kolkin, and K. Weinberger. 2015. "From word embeddings to document distances". In: *Proc. ICML*. 957–966.

Lafferty, J. and C. Zhai. 2001. "Document language models, query models, and risk minimization for information retrieval". In: *Proc. SIGIR*. ACM. 111–119.

Lagun, D., C.-H. Hsieh, D. Webster, and V. Navalpakkam. 2014. "Towards better measurement of attention and satisfaction in mobile search". In: *Proc. SIGIR*. ACM. 113–122.

Larsson, G., M. Maire, and G. Shakhnarovich. 2016. "Fractalnet: Ultra-deep neural networks without residuals". *arXiv preprint arXiv:1605.07648.*

Lavrenko, V. 2008. *A generative theory of relevance.* Vol. 26. Springer Science & Business Media.

Lavrenko, V. and W. B. Croft. 2001. "Relevance based language models". In: *Proc. SIGIR.* ACM. 120–127.

Le, H.-S., I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon. 2011. "Structured output layer neural network language model". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on.* IEEE. 5524–5527.

Le, Q. V. and T. Mikolov. 2014. "Distributed Representations of Sentences and Documents." In: *ICML.* Vol. 14. 1188–1196.

Lebret, R. and R. Collobert. 2013. "Word emdeddings through hellinger PCA". *arXiv preprint arXiv:1312.5542.*

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

LeCun, Y., F. J. Huang, and L. Bottou. 2004. "Learning methods for generic object recognition with invariance to pose and lighting". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on.* Vol. 2. IEEE. II–104.

LeCun, Y., K. Kavukcuoglu, and C. Farabet. 2010. "Convolutional networks and applications in vision". In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on.* IEEE. 253–256.

Levy, O. and Y. Goldberg. 2014. "Dependencybased word embeddings". In: *Proc. ACL.* Vol. 2. 302–308.

Levy, O., Y. Goldberg, and I. Dagan. 2015. "Improving distributional similarity with lessons learned from word embeddings". *Transactions of the Association for Computational Linguistics.* 3: 211–225.

Levy, O., Y. Goldberg, and I. Ramat-Gan. 2014. "Linguistic regularities in sparse and explicit word representations". *CoNLL-2014*: 171.

Levy, S. 2011. *In the plex: How Google thinks, works, and shapes our lives.* Simon and Schuster.

Li, H. and Z. Lu. "Deep Learning for Information Retrieval".

Li, H., J. Xu, *et al.* 2014. "Semantic matching in search". *Foundations and Trends® in Information Retrieval.* 7(5): 343–469.

Li, P., Q. Wu, and C. J. Burges. 2008. "Mcrank: Learning to rank using multiple classification and gradient boosting". In: *Advances in neural information processing systems.* 897–904.

Liebling, D. J., P. N. Bennett, and R. W. White. 2012. "Anticipatory search: using context to initiate search". In: *SIGIR.* ACM. 1035–1036.

Liu, C.-W., R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau. 2016. "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation". *arXiv preprint arXiv:1603.08023.*

Liu, T.-Y. 2009. "Learning to Rank for Information Retrieval". *Foundation and Trends in Information Retrieval.* 3(3): 225–331.

Liu, T.-Y., J. Xu, T. Qin, W. Xiong, and H. Li. 2007. "Letor: Benchmark dataset for research on learning to rank for information retrieval". In: *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval.* 3–10.

Liu, X., J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. "Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval". *Proc. NAACL, May 2015.*

Lu, Z. and H. Li. 2013. "A deep architecture for matching short texts". In: *Proc. NIPS.* 1367–1375.

Luce, R. D. 1959. *Individual Choice Behavior: A Theoretical Analysis.* John Wiley.

Lund, K. and C. Burgess. 1996. "Producing high-dimensional semantic spaces from lexical co-occurrence". *Behavior Research Methods, Instruments, & Computers.* 28(2): 203–208.

Luong, M.-T., H. Pham, and C. D. Manning. 2015. "Effective approaches to attention-based neural machine translation". *arXiv preprint arXiv:1508.04025.*

Luukkonen, P., M. Koskela, and P. Floréen. 2016. "Lstm-based predictions for proactive information retrieval". *arXiv preprint arXiv:1606.06137.*

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

Maaten, L. v. d. and G. Hinton. 2008. "Visualizing data using t-SNE". *Journal of Machine Learning Research*. 9(Nov): 2579–2605.

MacKay, D. J. and L. C. B. Peto. 1995. "A hierarchical Dirichlet language model". *Natural language engineering*. 1(3): 289–308.

Mallows, C. L. 1957. "Non-null ranking models. I". *Biometrika*. 44(1/2): 114–130.

Manning, C. 2016. "Understanding Human Language: Can NLP and Deep Learning Help?" In: *Proc. SIGIR*. ACM. 1–1.

Manning, C. D., P. Raghavan, H. Schütze, *et al.* 2008. *Introduction to information retrieval*. Vol. 1. Cambridge university press Cambridge.

Markovsky, I. 2011. *Low rank approximation: algorithms, implementation, applications*. Springer Science & Business Media.

Matveeva, I., C. Burges, T. Burkard, A. Laucius, and L. Wong. 2006. "High accuracy retrieval with multiple nested ranker". In: *Proc. SIGIR*. ACM. 437–444.

Metzler, D. and W. B. Croft. 2005. "A Markov random field model for term dependencies". In: *Proc. SIGIR*. ACM. 472–479.

Metzler, D., S. Dumais, and C. Meek. 2007. "Similarity measures for short segments of text". In: *European Conference on Information Retrieval*. Springer. 16–27.

Miao, J., J. X. Huang, and Z. Ye. 2012. "Proximity-based rocchio's model for pseudo relevance". In: *Proc. SIGIR*. ACM. 535–544.

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

Mikolov, T., M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur. 2010. "Recurrent neural network based language model." In: *Interspeech*. Vol. 2. 3.

Mikolov, T., S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur. 2011. "Extensions of recurrent neural network language model". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 5528–5531.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013b. "Distributed representations of words and phrases and their compositionality". In: *Proc. NIPS*. 3111–3119.

Mikolov, T., W.-t. Yih, and G. Zweig. 2013c. "Linguistic Regularities in Continuous Space Word Representations." In: *HLT-NAACL*. Citeseer. 746–751.

Mitra, B. 2015. "Exploring Session Context using Distributed Representations of Queries and Reformulations". In: *Proc. SIGIR*. ACM. 3–12.

Mitra, B. and N. Craswell. 2015. "Query Auto-Completion for Rare Prefixes". In: *Proc. CIKM*. ACM.

Mitra, B. and N. Craswell. 2017. "Neural Text Embeddings for Information Retrieval". In: *Proc. WSDM*. ACM. 813–814.

Mitra, B., F. Diaz, and N. Craswell. 2017a. "Learning to Match Using Local and Distributed Representations of Text for Web Search". In: *Proc. WWW*. 1291–1299.

Mitra, B., F. Diaz, and N. Craswell. 2017b. "Luandri: a Clean Lua Interface to the Indri Search Engine". In: *Proc. SIGIR*. ACM.

Mitra, B., E. Nalisnick, N. Craswell, and R. Caruana. 2016a. "A Dual Embedding Space Model for Document Ranking". *arXiv preprint arXiv:1602.01137*.

Mitra, B., M. Shokouhi, F. Radlinski, and K. Hofmann. 2014. "On User Interactions with Query Auto-Completion". In: *Proc. SIGIR*. 1055–1058.

Mitra, B., G. Simon, J. Gao, N. Craswell, and L. Deng. 2016b. "A Proposal for Evaluating Answer Distillation from Web Data". In: *Proceedings of the SIGIR 2016 WebQA Workshop*.

Mnih, A. and G. E. Hinton. 2009. "A scalable hierarchical distributed language model". In: *Advances in neural information processing systems*. 1081–1088.

Mnih, A. and Y. W. Teh. 2012. "A fast and simple algorithm for training neural probabilistic language models". *arXiv preprint arXiv:1206.6426*.

Mnih, V., N. Heess, A. Graves, and K. Kavukcuoglu. 2014. "Recurrent models of visual attention". In: *Proc. NIPS*. 2204–2212.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. "Human-level control through deep reinforcement learning". *Nature.* 518(7540): 529–533.

Montufar, G. F., R. Pascanu, K. Cho, and Y. Bengio. 2014. "On the number of linear regions of deep neural networks". In: *Proc. NIPS.* 2924–2932.

Morin, F. and Y. Bengio. 2005. "Hierarchical Probabilistic Neural Network Language Model." In: *Aistats.* Vol. 5. Citeseer. 246–252.

Nair, V. and G. E. Hinton. 2010. "Rectified linear units improve restricted boltzmann machines". In: *Proc. ICML.* 807–814.

Nalisnick, E., B. Mitra, N. Craswell, and R. Caruana. 2016. "Improving Document Ranking with Dual Word Embeddings". In: *Proc. WWW.*

Nanni, F., B. Mitra, M. Magnusson, and L. Dietz. 2017. "Benchmark for Complex Answer Retrieval". In: *Proc. ICTIR.* Amsterdam, The Netherlands: ACM.

Neubig, G., C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, *et al.* 2017. "DyNet: The Dynamic Neural Network Toolkit". *arXiv preprint arXiv:1701.03980.*

Nguyen, G.-H., L. Tamine, L. Soulier, and N. Bricon-Souf. 2016a. "Toward a deep neural approach for knowledge-based ir". *arXiv preprint arXiv:1606.07211.*

Nguyen, T., M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. 2016b. "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset". *arXiv preprint arXiv:1611.09268.*

Nogueira, R. and K. Cho. 2017. "Task-Oriented Query Reformulation with Reinforcement Learning". In: *Proc. EMNLP.* 574–583.

Oard, D. W. and A. R. Diekema. 1998. "Cross-language information retrieval." *Annual Review of Information Science and Technology (ARIST).* 33: 223–56.

Onal, K. D., Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, *et al.* 2018. "Neural information retrieval: At the end of the early years". *Information Retrieval Journal.* 21(2-3): 111–182.

Palangi, H., L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. 2014. "Semantic Modelling with Long-Short-Term Memory for Information Retrieval". *arXiv preprint arXiv:1412.6629.*

Palangi, H., L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. 2015. "Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval". *arXiv preprint arXiv:1502.06922.*

Pang, L., Y. Lan, J. Guo, J. Xu, and X. Cheng. 2016a. "A study of match-pyramid models on ad-hoc retrieval". *arXiv preprint arXiv:1606.04648.*

Pang, L., Y. Lan, J. Guo, J. Xu, S. Wan, and X. Cheng. 2016b. "Text Matching as Image Recognition". In: *Proc. AAAI.*

Pass, G., A. Chowdhury, and C. Torgeson. 2006. "A picture of search". In: *Proc. InfoScale.* ACM.

Pennington, J., R. Socher, and C. D. Manning. 2014. "Glove: Global vectors for word representation". *Proc. EMNLP.* 12: 1532–1543.

Plackett, R. L. 1975. "The analysis of permutations". *Applied Statistics*: 193–202.

Ponte, J. M. and W. B. Croft. 1998. "A language modeling approach to information retrieval". In: *Proc. SIGIR.* ACM. 275–281.

Rajaraman, A. 2008. "More data usually beats better algorithms". *Datawocky Blog.*

Rajpurkar, P., J. Zhang, K. Lopyrev, and P. Liang. 2016. "Squad: 100,000+ questions for machine comprehension of text". *arXiv preprint arXiv:1606.05250.*

Ranzato, M., C. Poultney, S. Chopra, and Y. LeCun. 2006. "Efficient learning of sparse representations with an energy-based model". In: *Proc. NIPS.* MIT Press. 1137–1144.

Rezende, D. J., S. Mohamed, and D. Wierstra. 2014. "Stochastic back-propagation and approximate inference in deep generative models". In: *Proc. ICML.*

Al-Rfou, R., G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, *et al.* 2016. "Theano: A Python framework for fast computation of mathematical expressions". *arXiv preprint arXiv:1605.02688.*

Richardson, M., C. J. Burges, and E. Renshaw. 2013. "MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text." In: *EMNLP*. Vol. 3. 4.

Robertson, S. E., E. Kanoulas, and E. Yilmaz. 2010. "Extending average precision to graded relevance judgments". In: *Proc. SIGIR*. ACM. 603–610.

Robertson, S. E., S. Walker, M. Beaulieu, M. Gatford, and A. Payne. 1996. "Okapi at TREC-4". In: *Proc. TREC*. Vol. 500. 73–97.

Robertson, S. and H. Zaragoza. 2009. "The probabilistic relevance framework: BM25 and beyond". *Foundations and Trends® in Information Retrieval.* 3(4): 333–389.

Robertson, S., H. Zaragoza, and M. Taylor. 2004. "Simple BM25 extension to multiple weighted fields". In: *Proc. CIKM*. ACM. 42–49.

Rohde, D. L., L. M. Gonnerman, and D. C. Plaut. 2006. "An improved model of semantic similarity based on lexical co-occurrence". *Communications of the ACM.* 8: 627–633.

Rong, X. 2014. "word2vec Parameter Learning Explained". *arXiv preprint arXiv:1411.2738.*

Rosset, C., D. Jose, G. Ghosh, B. Mitra, and S. Tiwary. 2018. "Optimizing Query Evaluations using Reinforcement Learning for Web Search". In: *Proc. SIGIR*. ACM.

Roy, D., D. Paul, M. Mitra, and U. Garain. 2016. "Using Word Embeddings for Automatic Query Expansion". *arXiv preprint arXiv:1606.07608.*

Rubner, Y., C. Tomasi, and L. J. Guibas. 1998. "A metric for distributions with applications to image databases". In: *Computer Vision, 1998. Sixth International Conference on.* IEEE. 59–66.

Ruder, S. 2016. "Approximating the Softmax for Learning Word Embeddings". Accessed June 7, 2017. URL: http://sebastianruder.com/word-embeddings-softmax/.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. 2015. "ImageNet Large Scale Visual Recognition Challenge". *International Journal of Computer Vision (IJCV)*. 115(3): 211–252. DOI: 10.1007/s11263-015-0816-y.

Sahlgren, M. 2006. "The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces". *PhD thesis*. Institutionen för lingvistik.

Sak, H., A. W. Senior, and F. Beaufays. 2014. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." In: *Interspeech*. 338–342.

Salakhutdinov, R. and G. Hinton. 2009. "Semantic hashing". *International Journal of Approximate Reasoning*. 50(7): 969–978.

Schmidhuber, J. 2015. "Deep learning in neural networks: An overview". *Neural networks*. 61: 85–117.

Schroff, F., D. Kalenichenko, and J. Philbin. 2015. "Facenet: A unified embedding for face recognition and clustering". In: *Proc. CVPR*. 815–823.

Senécal, J.-S. and Y. Bengio. 2003. "Adaptive importance sampling to accelerate training of a neural probabilistic language model". *Tech. rep.* Technical report, IDIAP.

Severyn, A. and A. Moschitti. 2015. "Learning to rank short text pairs with convolutional deep neural networks". In: *Proc. SIGIR*. ACM. 373–382.

Shan, Y., T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. 2016. "Deep Crossing: Web-scale modeling without manually crafted combinatorial features". In: *Proc. SIGKDD*. ACM. 255–262.

Shen, Y., X. He, J. Gao, L. Deng, and G. Mesnil. 2014a. "A latent semantic model with convolutional-pooling structure for information retrieval". In: *Proc. CIKM*. ACM. 101–110.

Shen, Y., X. He, J. Gao, L. Deng, and G. Mesnil. 2014b. "Learning semantic representations using convolutional neural networks for Web search". In: *Proc. WWW*. 373–374.

Shi, S., Q. Wang, P. Xu, and X. Chu. 2016. "Benchmarking State-of-the-Art Deep Learning Software Tools". *arXiv preprint arXiv:1608.07249.*

Shokouhi, M. and Q. Guo. 2015. "From queries to cards: Re-ranking proactive card recommendations based on reactive search history". In: *SIGIR*. ACM. 695–704.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.* 2016. "Mastering the game of Go with deep neural networks and tree search". *Nature*. 529(7587): 484–489.

Singhal, A., C. Buckley, and M. Mitra. 1996. "Pivoted document length normalization". In: *Proc. SIGIR*. ACM. 21–29.

Socher, R., C. C. Lin, C. Manning, and A. Y. Ng. 2011a. "Parsing natural scenes and natural language with recursive neural networks". In: *Proc. ICML*. 129–136.

Socher, R., J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011b. "Semi-supervised recursive autoencoders for predicting sentiment distributions". In: *Proc. EMNLP*. Association for Computational Linguistics. 151–161.

Song, Y. and Q. Guo. 2016. "Query-less: Predicting task repetition for nextgen proactive search and recommendation engines". In: *WWW*. International World Wide Web Conferences Steering Committee. 543–553.

Sordoni, A., Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie. 2015a. "A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion". *arXiv preprint arXiv:1507.02221*.

Sordoni, A., M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. 2015b. "A neural network approach to context-sensitive generation of conversational responses". In: *Proceedings of NAACL-HLT*. arXiv:1506.06714. 196–205.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. "Dropout: A simple way to prevent neural networks from overfitting". *The Journal of Machine Learning Research*. 15(1): 1929–1958.

Strohman, T., D. Metzler, H. Turtle, and W. B. Croft. 2005. "Indri: A language model-based search engine for complex queries". In: *Proceedings of the International Conference on Intelligent Analysis*. Vol. 2. No. 6. Citeseer. 2–6.

Sturm, B. L. 2014. "A simple method to determine if a music information retrieval system is a "horse"". *IEEE Transactions on Multimedia*. 16(6): 1636–1644.

Sukhbaatar, S., J. Weston, R. Fergus, *et al.* 2015. "End-to-end memory networks". In: *Proc. NIPS*. 2440–2448.

Sun, C., A. Shrivastava, S. Singh, and A. Gupta. 2017. "Revisiting unreasonable effectiveness of data in deep learning era". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 843–852.

Sun, F., J. Guo, Y. Lan, J. Xu, and X. Cheng. 2015. "Learning word representations by jointly modeling syntagmatic and paradigmatic relations". In: *Proc. ACL*.

Sun, F., J. Guo, Y. Lan, J. Xu, and X. Cheng. 2016a. "Semantic Regularities in Document Representations". *arXiv preprint arXiv:1603.07603*.

Sun, F., J. Guo, Y. Lan, J. Xu, and X. Cheng. 2016b. "Sparse word embeddings using l1 regularized online learning". In: *Proc. IJCAI*. 2915–2921.

Sutskever, I., J. Martens, and G. E. Hinton. 2011. "Generating text with recurrent neural networks". In: *Proc. ICML*. 1017–1024.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. "Going deeper with convolutions". In: *Proc. CVPR*. 1–9.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2013. "Intriguing properties of neural networks". *arXiv preprint arXiv:1312.6199*.

Tai, K. S., R. Socher, and C. D. Manning. 2015. "Improved semantic representations from tree-structured long short-term memory networks". *arXiv preprint arXiv:1503.00075*.

Taylor, M., H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. 2006. "Optimisation methods for ranking functions with multiple parameters". In: *Proc. CIKM*. ACM. 585–593.

Tishby, N., F. C. Pereira, and W. Bialek. 2000. "The information bottleneck method". *arXiv preprint physics/0004057*.

Tokui, S., K. Oono, S. Hido, and J. Clayton. 2015. "Chainer: a next-generation open source framework for deep learning". In: *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*.

Turney, P. D. and P. Pantel. 2010. "From frequency to meaning: Vector space models of semantics". *Journal of artificial intelligence research*. 37: 141–188.

Van Gysel, C., E. Kanoulas, and M. de Rijke. 2017a. "Pyndri: a Python Interface to the Indri Search Engine". *arXiv preprint arXiv:1701.00749*.

Van Gysel, C., B. Mitra, M. Venanzi, R. Rosemarin, G. Kukla, P. Grudzien, and N. Cancedda. 2017b. "Reply With: Proactive Recommendation of Email Attachments". In: *Proc. CIKM*.

Van Gysel, C., M. de Rijke, and E. Kanoulas. 2016a. "Learning latent vector spaces for product search". In: *Proc. CIKM*. ACM. 165–174.

Van Gysel, C., M. de Rijke, and M. Worring. 2016b. "Unsupervised, efficient and semantic expertise retrieval". In: *Proc. WWW*. International World Wide Web Conferences Steering Committee. 1069–1079.

Varma, M. and M. Cisse. 2015. "Extreme Classification 2015: The NIPS Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces". Accessed June 7, 2017. URL: http://manikvarma.org/events/XC15/.

Vaswani, A., Y. Zhao, V. Fossum, and D. Chiang. 2013. "Decoding with Large-Scale Neural Language Models Improves Translation." In: *EMNLP*. 1387–1392.

Vincent, P., A. de Brébisson, and X. Bouthillier. 2016. "Exact gradient updates in time independent of output size for the spherical loss family". *arXiv preprint arXiv:1606.08061*.

Vinyals, O., C. Blundell, T. Lillicrap, D. Wierstra, *et al.* 2016. "Matching networks for one shot learning". In: *Proc. NIPS*. 3630–3638.

Vinyals, O. and Q. Le. 2015. "A neural conversational model". *ICML Deep Learning Workshop*. arXiv:1506.05869.

Voorhees, E. M. and D. Harman. 2000. "Overview of the eighth text retrieval conference (TREC-8)": 1–24.

Voorhees, E. M., D. K. Harman, *et al.* 2005. *TREC: Experiment and evaluation in information retrieval*. Vol. 1. MIT press Cambridge.

Voorhees, E. M. and D. M. Tice. 2000. "Building a question answering test collection". In: *Proc. SIGIR*. ACM. 200–207.

Vulić, I. and M.-F. Moens. 2015. "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings". In: *Proc. SIGIR*. ACM. 363–372.

Wan, S., Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng. 2015. "A deep architecture for semantic matching with multiple positional sentence representations". *arXiv preprint arXiv:1511.08277*.

Wan, S., Y. Lan, J. Xu, J. Guo, L. Pang, and X. Cheng. 2016. "Match-srnn: Modeling the recursive matching structure with spatial rnn". *arXiv preprint arXiv:1604.04378*.

Wan, X. 2007. "A novel document similarity measure based on earth mover's distance". *Information Sciences*. 177(18): 3718–3730.

Wan, X. and Y. Peng. 2005. "The earth mover's distance as a semantic measure for document similarity". In: *Proc. CIKM*. ACM. 301–302.

Wang, J., L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. 2017. "IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models". *arXiv preprint arXiv:1705.10513*.

Wang, L., J. Lin, and D. Metzler. 2011. "A cascade ranking model for efficient ranked retrieval". In: *Proc. SIGIR*. ACM. 105–114.

Weston, J., A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. 2015. "Towards ai-complete question answering: A set of prerequisite toy tasks". *arXiv preprint arXiv:1502.05698*.

Weston, J., S. Chopra, and A. Bordes. 2014. "Memory networks". *arXiv preprint arXiv:1410.3916*.

White, R. W. 2016. *Interactions with search systems*. Cambridge University Press.

Wissner-Gross, A. 2016. "Datasets Over Algorithms". URL: https://www.edge.org/response-detail/26587.

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

Xia, F., T.-Y. Liu, J. Wang, W. Zhang, and H. Li. 2008. "Listwise approach to learning to rank: theory and algorithm". In: *Proc. ICML*. ACM. 1192–1199.

Xia, L., J. Xu, Y. Lan, J. Guo, and X. Cheng. 2016. "Modeling Document Novelty with Neural Tensor Network for Search Result Diversification". In: *Proc. SIGIR*. ACM. 395–404.

Xie, Y. and D. O'Hallaron. 2002. "Locality in search engine queries and its implications for caching". In: *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. Vol. 3. IEEE. 1238–1247.

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. 2015. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2048–2057.

Yan, R., Y. Song, and H. Wu. 2016. "Learning to respond with deep neural networks for retrieval-based human-computer conversation system". In: *Proc. SIGIR*. ACM. 55–64.

Yan, X., J. Guo, S. Liu, X. Cheng, and Y. Wang. 2013. "Learning topics in short texts by non-negative matrix factorization on term correlation matrix". In: *Proceedings of the SIAM International Conference on Data Mining*.

Yang, L., Q. Ai, J. Guo, and W. B. Croft. 2016. "aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model". In: *Proc. CIKM*. ACM. 287–296.

Yang, Y., W.-t. Yih, and C. Meek. 2015. "WikiQA: A Challenge Dataset for Open-Domain Question Answering." In: *EMNLP*. Citeseer. 2013–2018.

Yih, W.-t., K. Toutanova, J. C. Platt, and C. Meek. 2011. "Learning discriminative projections for text similarity measures". In: *Proc. CoNLL*. Association for Computational Linguistics. 247–256.

Yin, W., H. Schütze, B. Xiang, and B. Zhou. 2015. "Abcnn: Attention-based convolutional neural network for modeling sentence pairs". *arXiv preprint arXiv:1512.05193*.

Yu, D., A. Eversole, M. Seltzer, K. Yao, Z. Huang, B. Guenter, O. Kuchaiev, Y. Zhang, F. Seide, H. Wang, *et al.* "An introduction to computational networks and the computational network toolkit".

Yue, Y. and C. Burges. 2007. "On using simultaneous perturbation stochastic approximation for IR measures, and the empirical optimality of LambdaRank". In: *NIPS Machine Learning for Web Search Workshop.*

Yue, Y., T. Finley, F. Radlinski, and T. Joachims. 2007. "A support vector method for optimizing average precision". In: *Proc. SIGIR.* ACM. 271–278.

Zamani, H. and W. B. Croft. 2016a. "Embedding-based query language models". In: *Proc. ICTIR.* ACM. 147–156.

Zamani, H. and W. B. Croft. 2016b. "Estimating embedding vectors for queries". In: *Proc. ICTIR.* ACM. 123–132.

Zamani, H., B. Mitra, X. Song, N. Craswell, and S. Tiwary. 2018. "Neural Ranking Models with Multiple Document Fields". In: *Proc. WSDM.*

Zaragoza, H., N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson. 2004. "Microsoft Cambridge at TREC 13: Web and Hard Tracks." In: *TREC.* Vol. 4. 1–1.

Zhai, C. and J. Lafferty. 2001. "A study of smoothing methods for language models applied to ad hoc information retrieval". In: *Proc. SIGIR.* ACM. 334–342.

Zhang, C., S. Bengio, M. Hardt, B. Recht, and O. Vinyals. 2016. "Understanding deep learning requires rethinking generalization". *arXiv preprint arXiv:1611.03530.*

Zhao, H., Z. Lu, and P. Poupart. 2015. "Self-Adaptive Hierarchical Sentence Model". *arXiv preprint arXiv:1504.05070.*

Zheng, G. and J. Callan. 2015. "Learning to Reweight Terms with Distributed Representations". In: *Proc. SIGIR.* ACM. 575–584.

Zhou, X., D. Dong, H. Wu, S. Zhao, R. Yan, D. Yu, X. Liu, and H. Tian. 2016. "Multi-view response selection for human-computer conversation". *EMNLP'16.*

Zhu, M. 2004. "Recall, precision and average precision". *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo.* 2: 30.

Zoph, B., A. Vaswani, J. May, and K. Knight. 2016. "Simple, fast noise-contrastive estimation for large rnn vocabularies". In: *Proceedings of NAACL-HLT.* 1217–1222.

Zuccon, G., B. Koopman, P. Bruza, and L. Azzopardi. 2015. "Integrating
    and evaluating neural word embeddings in information retrieval".
    In: *Proc. ADCS*. ACM. 12.

Zweig, G. and K. Makarychev. 2013. "Speed regularization and optimal-
    ity in word classing". In: *Acoustics, Speech and Signal Processing
    (ICASSP), 2013 IEEE International Conference on*. IEEE. 8237–
    8241.