9

Selected Applications in Information Retrieval

9.1 A brief introduction to information retrieval

Information retrieval (IR) is a process whereby a user enters a query into the automated computer system that contains a collection of many documents with the goal of obtaining a set of most relevant documents. Queries are formal statements of information needs, such as search strings in web search engines. In IR, a query does not uniquely identify a single document in the collection. Instead, several documents may match the query with different degrees of relevancy.

A document, sometimes called an object as a more general term which may include not only a text document but also an image, audio (music or speech), or video, is an entity that contains information and represented as an entry in a database. In this section, we limit the "object" to only text documents. User queries in IR are matched against the documents' representation stored in the database. Documents themselves often are not kept or stored directly in the IR system. Rather, they are represented in the system by metadata. Typical IR systems compute a numeric score on how well each document in the database matches the query, and rank the objects according to this value. The top-ranking documents from the system are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Based partly on [236], common IR methods consist of several categories:

- Boolean retrieval, where a document either matches a query or does not.
- Algebraic approaches to retrieval, where models are used to represent documents and queries as vectors, matrices, or tuples. The similarity of the query vector and document vector is represented as a scalar value. This value can be used to produce a list of documents that are rank-ordered for a query. Common models and methods include vector space model, topic-based vector space model, extended Boolean model, and latent semantic analysis.
- Probabilistic approaches to retrieval, where the process of IR is treated as a probabilistic inference. Similarities are computed as probabilities that a document is relevant for a given query, and the probability value is then used as the score in ranking documents. Common models and methods include binary independence model, probabilistic relevance model with the BM25 relevance function, methods of inference with uncertainty, probabilistic, language modeling, http://en.wikipedia.org/wiki/Uncertain_inference and the technique of latent Dirichlet allocation.
- Feature-based approaches to retrieval, where documents are viewed as vectors of values of feature functions. Principled methods of "learning to rank" are devised to combine these features into a single relevance score. Feature functions are arbitrary functions of document and query, and as such Feature-based approaches can easily incorporate almost any other retrieval model as just yet another feature.

Deep learning applications to IR are rather recent. The approaches in the literature so far belong mostly to the category of feature-based approaches. The use of deep networks is mainly for extracting semantically meaningful features for subsequent document ranking stages.

We will review selected studies in the recent literature in the remainder of this section below.

9.2 Semantic hashing with deep autoencoders for document indexing and retrieval

Here we discuss the "semantic hashing" approach for the application of deep autoencoders to document indexing and retrieval as published in [159, 314]. It is shown that the hidden variables in the final layer of a DBN not only are easy to infer after using an approximation based on feed-forward propagation, but they also give a better representation of each document, based on the word-count features, than the widely used latent semantic analysis and the traditional TF-IDF approach for information retrieval. Using the compact code produced by deep autoencoders, documents are mapped to memory addresses in such a way that semantically similar text documents are located at nearby addresses to facilitate rapid document retrieval. The mapping from a word-count vector to its compact code is highly efficient, requiring only a matrix multiplication and a subsequent sigmoid function evaluation for each hidden layer in the encoder part of the network.

A deep generative model of DBN is exploited for the above purpose as discussed in [165]. Briefly, the lowest layer of the DBN represents the word-count vector of a document and the top layer represents a learned binary code for that document. The top two layers of the DBN form an undirected associative memory and the remaining layers form a Bayesian (also called belief) network with directed, top-down connections. This DBN, composed of a set of stacked RBMs as we reviewed in Section 5, produces a feed-forward "encoder" network that converts word-count vectors to compact codes. By composing the RBMs in the opposite order, a "decoder" network is constructed that maps compact code vectors into reconstructed word-count vectors. Combining the encoder and decoder, one obtains a deep autoencoder (subject to further fine-tuning as discussed in Section 4) for document coding and subsequent retrieval.

After the deep model is trained, the retrieval process starts with mapping each query into a 128-bit binary code by performing a forward pass through the model with thresholding. Then the Hamming distance between the query binary code and all the documents' 128-bit binary codes, especially those of the "neighboring" documents defined in the semantic space, are computed extremely efficiently. The efficiency is accomplished by looking up the neighboring bit vectors in the hash table. The same idea as discussed here for coding text documents for information retrieval has been explored for audio document retrieval and speech feature coding problems with some initial exploration reported in [100], discussed in Section 4 in detail.

9.3 Deep-structured semantic modeling (DSSM) for document retrieval

Here we discuss the more advanced and recent approach to large-scale document retrieval (Web search) based on a specialized deep architecture, called deep-structured semantic model or deep semantic similarity model (DSSM), as published in [172], and its convolutional version (C-DSSM), as published in [328].

Modern search engines retrieve Web documents mainly by matching keywords in documents with those in a search query. However, lexical matching can be inaccurate due to the fact that a concept is often expressed using different vocabularies and language styles in documents and queries. Latent semantic models are able to map a query to its relevant documents at the semantic level where lexical-matching often fails [236]. These models address the language discrepancy between Web documents and search queries by grouping different terms that occur in a similar context into the same semantic cluster. Thus, a query and a document, represented as two vectors in the lower-dimensional semantic space, can still have a high similarity even if they do not share any term. Probabilistic topic models such as probabilistic latent semantic models and latent Dirichlet allocation models have been proposed for semantic matching to partially overcome such difficulties. However, the improvement on IR tasks has not been as significant as originally expected because of two main factors: (1) most state-of-the-art latent semantic models are based on linear projection, and thus are inadequate in capturing effectively the complex semantic properties of documents;

and (2) these models are often trained in an unsupervised manner using an objective function that is only loosely coupled with the evaluation metric for the retrieval task. In order to improve semantic matching for IR, two lines of research have been conducted to extend the above latent semantic models. The first is the semantic hashing approach reviewed in Section 9.1 above in this section based on the use of deep autoencoders [165, 314]. While the hierarchical semantic structure embedded in the query and the document can be extracted via deep learning, the deep learning approach used for their models still adopts an unsupervised learning method where the model parameters are optimized for the re-construction of the documents rather than for differentiating the relevant documents from the irrelevant ones for a given query. As a result, the deep neural network models do not significantly outperform strong baseline IR models that are based on lexical matching. In the second line of research, click-through data, which consists of a list of queries and the corresponding clicked documents, is exploited for semantic modeling so as to bridge the language discrepancy between search queries and Web documents in recent studies [120, 124]. These models are trained on click-through data using objectives that tailor to the document ranking task. However, these click-through-based models are still linear, suffering from the issue of expressiveness. As a result, these models need to be combined with the keyword matching models (such as BM25) in order to obtain a significantly better performance than baselines.

The DSSM approach reported in [172] aims to combine the strengths of the above two lines of work while overcoming their weaknesses. It uses the DNN architecture to capture complex semantic properties of the query and the document, and to rank a set of documents for a given query. Briefly, a nonlinear projection is performed first to map the query and the documents to a common semantic space. Then, the relevance of each document given the query is calculated as the cosine similarity between their vectors in that semantic space. The DNNs are trained using the click-through data such that the conditional likelihood of the clicked document given the query is maximized. Different from the previous latent semantic models that are learned in an unsupervised fashion, the DSSM is optimized directly for Web

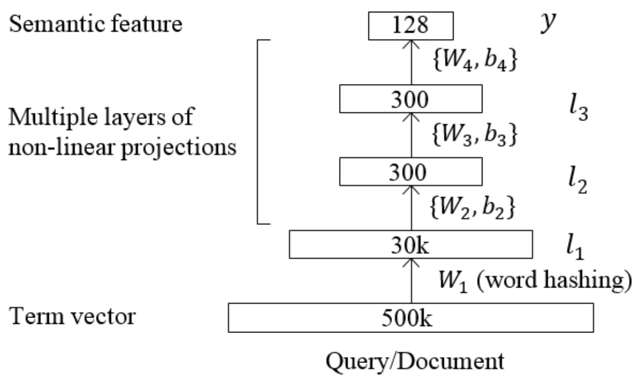

Figure 9.1: The DNN component of the DSSM architecture for computing semantic features. The DNN uses multiple layers to map high-dimensional sparse text features, for both Queries and Documents into low-dimensional dense features in a semantic space. [after [172], @CIKM].

document ranking, and thus gives superior performance. Furthermore, to deal with large vocabularies in Web search applications, a new *word hashing* method is developed, through which the high-dimensional term vectors of queries or documents are projected to low-dimensional letter based n-gram vectors with little information loss.

Figure 9.1 illustrates the DNN part in the DSSM architecture. The DNN is used to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

To show the computational steps in the various layers of the DNN in Figure 9.1, we denote x as the input term vector, y as the output vector, l_i, $i = 1, \ldots, N-1$, as the intermediate hidden layers, W_i as the ith projection matrix, and b_i as the ith bias vector, we have

$$\begin{aligned} l_1 &= W_1 x, \\ l_i &= f(W_i l_{i-1} + b_i), \quad i > 1 \\ y &= f(W_N l_{N-1} + b_N), \end{aligned}$$

where *tanh* function is used at the output layer and the hidden layers $l_i, i = 2, \ldots, N-1$:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}.$$

The semantic relevance score between a query Q and a document D can then be computed as the consine distance

$$R(Q, D) = \text{cosine}(y_Q, y_D) = \frac{y_Q^T y_D}{\|y_Q\| \|y_D\|},$$

where y_Q and y_D are the concept vectors of the query and the document, respectively. In Web search, given the query, the documents can be sorted by their semantic relevance scores.

Learning of the DNN weights W_i and b_i shown in Figure 9.1 is an important contribution of the study of [172]. Compared with the DNNs used in speech recognition where the targets or labels of the training data are readily available, the DNN in the DSSM does not have such label information well defined. That is, rather than using the common cross entropy or mean square errors as the training objective function, IR-centric loss functions need to be developed in order to train the DNN weights in the DSSM using the available data such as click-through logs.

The click-through logs consist of a list of queries and their clicked documents. A query is typically more relevant to the documents that are clicked on than those that are not. This weak supervision information can be exploited to train the DSSM. More specifically, the weight matrices in the DSSM, W_i, is learned to maximize the posterior probability of the clicked documents given the queries

$$P(D \mid Q) = \frac{\exp(\gamma R(Q, D))}{\sum_{D' \in D} \exp(\gamma R(Q, D'))}$$

defined on the semantic relevance score $R(Q, D)$ between the Query (Q) and the Document (D), where γ is a smoothing factor set empirically on a held-out data set, and **D** denotes the set of candidate documents to be ranked. Ideally, **D** should contain all possible documents, as in the maximum mutual information training for speech recognition where all possible negative candidates may be considered [147]. However in this case $\mathbf{D}$ is of Web scale and thus is intractable in practice. In the implementation of DSSM learning described in [172], a subset of the negative candidates are used, following the common practice adopted in MCE (Minimum Classification Error) training in speech recognition [52, 118, 417, 418]. In other words, for each query and clicked-document pair, denoted by (QD^+) where Q is a query and D^+ is the clicked document, the set of $\boldsymbol{D}$ is approximated by including D^+ and only four randomly selected unclicked documents, denoted by $D_j^-; j = 1, \ldots, 4\}$. In the study reported in [172], no significant difference was found when different sampling strategies were used to select the unclicked documents.

With the above simplification the DSSM parameters are estimated to maximize the approximate likelihood of the clicked documents given the queries across the training set

$$L(\Lambda) = \log \prod_{(Q, D^+, D_j^-)} P(D^+ \mid Q),$$

where Λ denotes the parameter set of the DNN weights $\{W_i\}$ in the DSSM. In Figure 9.2, we show the overall DSSM architecture that contains several DNNs. All these DNNs share the same weights but take different documents (one positive and several negatives) as inputs when training the DSSM parameters. Details of the gradient computation of this approximate loss function with respect to the DNN weights tied across documents and queries can be found in [172] and are not elaborated here.

Most recently, the DSSM described above has been extended to its convolutional version, or C-DSSM [328]. In the C-DSSM, semantically similar words within context are projected to vectors that are close to each other in the contextual feature space through a convolutional structure. The overall semantic meaning of a sentence is found to be determined by a few *key* words in the sentence, and thus the C-DSSM uses an additional max pooling layer to extract the most salient local features to form a fixed-length global feature vector. The global feature vector is then fed to the remaining nonlinear DNN layer(s) to map it to a point in the shared semantic space.

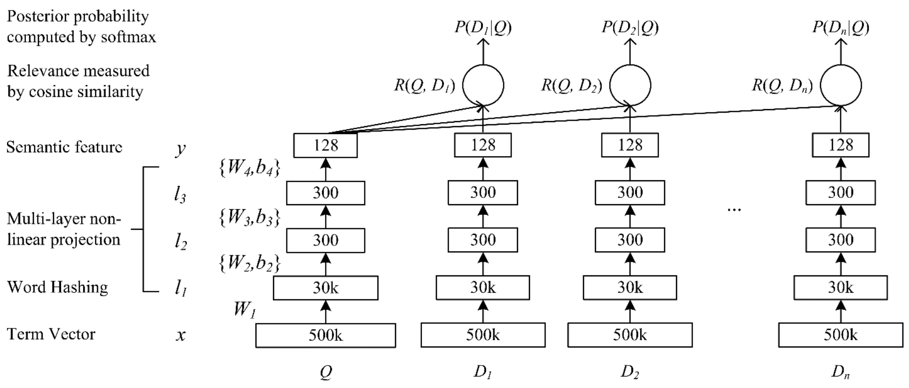

Figure 9.2: Architectural illustration of the DSSM for document retrieval (from [170, 171]). All DNNs shown have shared weights. A set of n documents are shown here to illustrate the random negative sampling discussed in the text for simplifying the training procedure for the DSSM. [after [172], @CIKM].

The convolutional neural network component of the C-DSSM is shown in Figure 9.3, where a window size of three is illustrated for the convolutional layer. The overall C-DSSM architecture is similar to the DSSM architecture shown in Figure 9.2 except that the fully-connected DNNs are replaced by the convolutional neural networks with locally-connected tied weights and additional max-pooling layers. The model component shown in Figure 9.3 contains (1) a word hashing layer to transform words into letter-tri-gram count vectors in the same way as the DSSM; (2) a convolutional layer to extract local contextual features for each context window; (3) a max-pooling layer to extract and combine salient local contextual features to form a global feature vector; and (4) a semantic layer to represent the high-level semantic information of the input word sequence.

The main motivation for using the convolutional structure in the C-DSSM is its ability to map a variable-length word sequence to a low-dimensional vector in a latent semantic space. Unlike most previous models that treat a query or a document as a bag of words, a query or a document in the C-DSSM is viewed as a sequence of words with contextual structures. By using the convolutional structure, local contextual information at the word n-gram level is modeled first. Then,

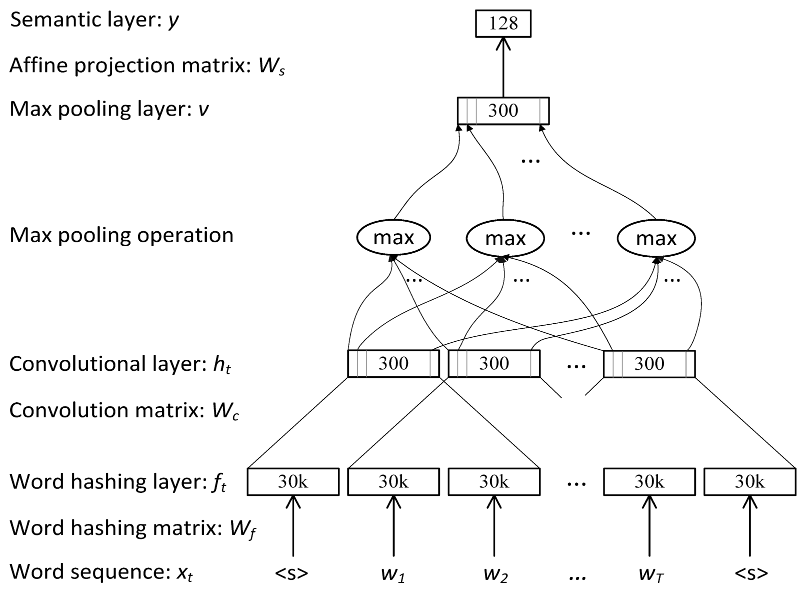
Semantic layer: y
Affine projection matrix: W_s
Max pooling layer: v
Max pooling operation
Convolutional layer: h_t
Convolution matrix: W_c
Word hashing layer: f_t
Word hashing matrix: W_f
Word sequence: x_t
128
300
max
30k
<s>
w_1
w_2
w_T

Figure 9.3: The convolutional neural network component of the C-DSSM, with the window size of three is illustrated for the convolutional layer. [after [328], @WWW].

salient local features in a word sequence are combined to form a global feature vector. Finally, the high-level semantic information of the word sequence is extracted to form a global vector representation. Like the DSSM just described, the C-DSSM is also trained on click-through data by maximizing the conditional likelihood of the clicked documents given a query using the back-propagation algorithm.

9.4 Use of deep stacking networks for information retrieval

In parallel with the IR studies reviewed above, the deep stacking network (DSN) discussed in Section 6 has also been explored recently for IR with insightful results [88]. The experimental results suggest that the classification error rate using the binary decision of "relevant" versus "non-relevant" from the DSN, which is closely correlated with the DSN training objective, is also generally correlated well with the NDCG (normalized discounted cumulative gain) as the most common

IR quality measure. The exception is found in the region of high IR quality.

As described in Section 6, the simplicity of the DSN's training objective, the mean square error (MSE), drastically facilitates its successful applications to image recognition, speech recognition, and speech understanding. The MSE objective and classification error rate have been shown to be well correlated in these speech or image applications. For information retrieval (IR) applications, however, the inconsistency between the MSE objective and the desired objective (e.g., NDCG) is much greater than that for the above classification-focused applications. For example, the NDCG as a desirable IR objective function is a highly non-smooth function of the parameters to be learned, with a very different nature from the nonlinear relationship between MSE and classification error rate. Thus, it is of interest to understand to what extent the NDCG is reasonably well correlated with classification rate or MSE where the relevance level in IR is used as the DSN prediction target. Further, can the advantage of learning simplicity in the DSN be applied to improve IR quality measures such as the NDCG? Our experimental results presented in [88] provide largely positive answers to both of the above questions. In addition, special care that need to be taken in implementing DSN learning algorithms when moving from classification to IR applications are addressed.

The IR task in the experiments of [88] is the sponsored search related to ad placement. In addition to the organic web search results, commercial search engines also provide supplementary sponsored results in response to the user's query. The sponsored search results are selected from a database pooled by advertisers who bid to have their ads displayed on the search result pages. Given an input query, the search engine will retrieve relevant ads from the database, rank them, and display them at the proper place on the search result page; e.g., at the top or right hand side of the web search results. Finding relevant ads to a query is quite similar to common web search. For instance, although the documents come from a constrained database, the task resembles typical search ranking that targets on predicting document relevance to the input query. The experiments conducted for this task are the first with the use of deep learning techniques (based on the DSN architecture) on the ad-related IR problem. The preliminary results from the experiments are the close correlation between the MSE as the DSN training objective with the NDCG as the IR quality measure over a wide NDCG range.