Neural Networks for Information Retrieval

Tom Kenter∗
University of Amsterdam
Amsterdam, The Netherlands
tom.kenter@uva.nl

Alexey Borisov
Yandex
Moscow, Russia
alborisov@yandex-team.ru

Christophe Van Gysel
University of Amsterdam
Amsterdam, The Netherlands
cvangysel@uva.nl

Mostafa Dehghani
University of Amsterdam
Amsterdam, The Netherlands
dehghani@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

Bhaskar Mitra
Microsoft, University College London
Cambridge, UK
bmitra@microsoft.com

CCS CONCEPTS
• Information systems → Personalization; Probabilistic retrieval models; Language models; Learning to rank; Question answering;

EXTENDED ABSTRACT

Machine learning plays a role in many aspects of modern IR systems, and deep learning is applied in all of them. The fast pace of modern-day research has given rise to many different approaches for many different IR problems. The amount of information available can be overwhelming both for junior students and for experienced researchers looking for new research topics and directions. Additionally, it is interesting to see what key insights into IR problems the new technologies are able to give us. The aim of this full-day tutorial is to give a clear overview of current tried-and-trusted neural methods in IR and how they benefit IR research. It covers key architectures, as well as the most promising future directions.

1 MOTIVATION

Prompted by the advances of deep learning in computer vision research, neural networks have resurfaced as a popular machine learning paradigm in many other directions of research as well, including information retrieval. Recent years have seen neural networks being applied to all key parts of the typical modern IR pipeline, such as a label for learning task-specific latent representations [25–37], semantic matching settings [38], sequence-to-sequence models [40], dropout [41], sequence, many different architectures and paradigms have been proposed, such as auto-encoders, recursive networks, recurrent networks, convolutional networks, various embedding methods, deep reinforcement and deep q-learning, and, more recently, generative adversarial networks, of which most have been applied in IR settings. The aim of the neural networks for IR (NN4IR) tutorial is to provide a clear overview of the main network architectures currently applied in IR and to show explicitly how they relate to previous work. The tutorial covers methods applied in industry and academia, with in-depth insights into the underlying theory, core IR tasks, applicability, key assets and handicaps, scalability concerns and practical tips and tricks.

We expect the tutorial to be useful both for academic and industrial researchers and practitioners who either want to develop new neural models, use them in their own research in other areas or apply the models described here to improve actual IR systems.

2 OBJECTIVES

The material in the tutorial covers a broad range of IR applications. It is structured as follows:

Preliminaries (60 minutes). The recent surge of interest in deep learning has given rise to a myriad of architectures. Different though the inner structures of neural networks can be, there are many concepts common to all of them. This first session covers the preliminaries; we briefly recapitulate the basic concepts involved in neural systems, such as back propagation [42], distributed representations/embodiments [39], convolutional layers [30], recurrent networks [38], sequence-to-sequence models [50], dropout [49], loss functions, optimization schemes like Adam [29].

Semantic matching I: supervised learning (60 minutes). The problem of matching items based on their textual descriptions arises in many IR systems. The traditional approach involves counting query term occurrences in the description text (e.g., BM25 [43]). However, to bridge the lexical gap caused by vocabulary-related and linguistic differences many latent semantic models have been proposed [6, 13, 24, 55], and more recently neural embedding methods [39]. In this session we will focus on semantic matching settings where a supervised signal is available. The signal can be explicit, such as a label for learning task-specific latent representations [25–27, 36, 47, 48], or relevance labels and, more implicitly, clicks for neural IR methods [15, 19, 31, 41, 42].
Semantic matching II: Semi- and unsupervised learning (60 minutes). How to learn semantics in the absence of relevance labels or user interaction signals? Depending on the available resources, one can choose for semi- or unsupervised matching models.

Unsupervised semantic matching methods can be categorized into two groups. First, using pre-trained word embeddings like combining traditional retrieval models with an embedding-based translation model [16, 58], using pre-trained embeddings for query expansion to improve retrieval [57], and representing documents as Bag-of-Word-Embeddings (BoWE) [20, 27]. Second, learning representations from scratch like learning representations of words and documents [28, 32] and employing them in retrieval task [2, 3], and learning representations in an end-to-end neural model for learning a specific task like entity ranking for expert finding [53] or product search [52].

In semi-supervised learning, on the other hand, queries (without relevance labels), or prior knowledge about document similarity can be used to induce pseudo-relevance labels. Furthermore, it is possible to use heuristic methods to generate weak supervision signals and to go beyond them by employing proper learning objectives and network designs [14].

Learning to rank (45 minutes). Capturing the notion of relevance for ranking needs to account for different aspects of the query, the document, and their relationship. Neural methods for ranking can use both manually crafted features from query and document and combine them with regards to a ranking objective, or learn latent representations for them in situ.

Irrespective of how the query and the documents are featureized, a neural learning to rank model can be designed for different scenarios, each having its own appropriate loss function. An example is the point-wise versus pair-wise paradigm, each of which has a different objective that calibrates either scores or the relative ranking of documents, given a query. Neural learning to rank models can also be designed to be provided with different levels of supervision during training—unsupervised [45, 52, 53], semi/weakly-supervised [14, 51], or fully-supervised using labeled [42] or click data [26].

Modeling user behavior (45 minutes). Modeling user browsing behavior plays an important role in the development of modern IR systems. Accurately interpreting user clicks is difficult due to various types of bias. For example, users tend to click more on results ranked on top positions (position bias) and visually salient results (attention bias). The traditional way to account for these biases is to design a Probabilistic Graphical Model (PGM) that explains relationships between click/skip events (observed variables) and examination (unobserved variables). Over the last decade many PGM-based click models have been proposed (see [12] for an overview).

However, these click models can model only those patterns that are explicitly encoded in their PGMs. Recently, it was shown that recurrent neural networks can learn to account for biases in user clicks directly from the click-through data, i.e., without the need for a predefined set of rules as is customary for PGM-based click models [9]. Additionally, there are similar biases in click dwell times, which the neural approach can account for too.

Generating responses (45 minutes). Recent inventions such as smart home devices, voice search and virtual assistants provide new ways of accessing information. They require a different response format than the classic ten blue links. Targeting this newly emerging demand, some models have been proposed to respond by generating natural language replies on the fly, rather than by (re)ranking a fixed set of items or extracting passages from existing pages.

Examples are conversational and dialog systems [7, 34, 54] or machine reading and question answering tasks where the model either infers the answer from unstructured data, like textual documents that do not necessarily feature the answer literally [21, 22, 46, 56], or generates natural language given structured data, like data from knowledge graphs or from external memories [1, 18, 33, 37, 40].

Outlook (30 minutes). In this session, open research questions and future directions are discussed. One of the big challenges for IR at the moment is how to process full document text using neural networks. On a higher level, it is probably desirable to learn all components of a full IR system in an end-to-end fashion.

Another challenge is maintaining long term (multiple day) search sessions or conversations. Which naturally leads to an additional open problem: how to evaluate (neural) conversational systems.

Finally, we cover recent advances, like Generative Adversarial Networks [17].

Summing up, the objectives of the NN4IR tutorial are as follows:

- Give an extensive overview of neural network architectures currently employed in IR, both in academia and industry.
- Provide theoretical background, thereby equipping participants with the necessary means to form intuitions about various neural methods and their applicability.
- Identify the IR lessons learned by employing neural methods.
- Give practical tips and tricks, regarding network design, optimization, hyperparameter values, based on industry best practice.
- Discuss promising future research directions.

The target audience consists of researchers and developers in information retrieval who are interested in gaining an in-depth understanding of neural models across a wide range of IR problems. The tutorial will be useful as an overview for anyone new to the deep learning field as well as for practitioners seeking concrete recipes. The tutorial aims to provide a map of the increasingly rich landscape of neural models in IR.

By the end of the tutorial, attendees will be familiar with the main architectures of neural networks as applied in IR and they will have informed intuitions of their key properties and of the insights they bring into core IR problems. We aim to provide an overview of the main directions currently employed, together with a clear understanding of the underlying theory and insights, illustrated with examples.

3 FORMAT AND DETAILED SCHEDULE

Table 1 gives an overview of the time schedule of the tutorial. Below we provide the details for each session.

We bring a team team of six lecturers, all with their specific areas of specialization. Each session will have two expert lecturers who will together present the session. The initials below refer to the lecturers for this tutorial.

Preliminaries — 60 minutes (TK, MdR).
Semi-supervised semantic matching – We cover how to
10 mins

Short text similarity – Given two short texts, e.g., queries
25 mins

Learning unsupervised representations from scratch for
25 mins

Unsupervised semantic matching using pre-trained word
15 mins

Learning to match using local representations – Use both
15 mins

Word embeddings for matching – Learning embeddings
10 mins

Convolutional networks – CNNs are primarily employed
20 mins

Embedding methods – We detail how word2vec works
10 mins

Recurrent neural networks – We cover the basics, based
10 mins

Sequence-to-sequence models – Basic architecture of
10 mins

Semantic matching I: supervised — 60 minutes (AB, BM).

10 mins Back propagation – Given a standard feedforward net-
work, we show the math and the intuition of back propagation. Also we will briefly touch on dropout.

5 mins Distributed representations – We show what a distributed representation is, and how distributed representations can be used.

10 mins Recurrent neural networks – We cover the basics, based on a language modeling scenario, including LSTMs [23] and GRUs [11].

10 mins Embedding methods – We detail how word2vec works and how it can be applied to different settings.

10 mins Sequence-to-sequence models – Basic architecture of seq2seq models [50], including the attention mechanism [4].

10 mins Convolutional networks – CNNs are primarily employed in computer vision, but can be beneficial in text classification tasks too.

5 mins Optimisation schemes – Standard back propagation with a fixed learning rate is typically replaced by more sophisticated schemes that handle learning rate annealing.

Semantic matching II: Semi- and unsupervised semantic matching — 60 minutes (MD, CVG).

10 mins Short text similarity – Given two short texts, e.g., queries or sentences, how can we predict if they are semantically similar?

15 mins Word embeddings for matching – Learning embeddings from click data.

20 mins Deep neural architectures for matching – Deep Structured Semantic Model (DSSM) [26].

15 mins Learning to match using local representations – Use both local and global representations for query-document matching.

Semantic matching II: Semi- and unsupervised semantic matching — 60 minutes (MD, CVG).

10 mins Semi-supervised semantic matching – We cover how to model pseudo-labeling using prior knowledge like document similarity, or by employing heuristic methods as weak supervision signals.

25 mins Unsupervised semantic matching using pre-trained word embeddings – We show how different IR tasks benefit from using pre-trained word embeddings by pre-estimating representations for query and documents, or as warm start for representation learning during training.

25 mins Learning unsupervised representations from scratch for semantic matching – We explain how to learn representations of words and documents in an unsupervised manner without any relevance label, that satisfy the requirements of IR problems.

Learning to rank, 45 minutes (CVG, BM).

10 mins Feature-based models for representation learning – We explain how to train a ranker using featurized input, and how to feed the network with raw data to have it learn representations jointly with a downstream task.

15 mins Ranking objectives and loss functions – We describe point-wise and pair-wise settings for the ranking task and the proper loss functions for each setting.

20 mins Training under different levels of supervision – We cover how to train a neural ranker in an unsupervised way, with weak, semi- or full supervision and discuss requirements and concerns of each situation.

Modeling user behavior — 45 minutes (AB, MdR).

10 mins Biases and PGM-based click models – We introduce notions of bias in user behavior and explain how to account for them using probabilistic graphical models (PGMs).

25 mins Neural click models – We discuss weaknesses of PGM-based approaches and present an alternative based on recurrent neural networks.

10 mins Hybrid approach – We describe recent work on modeling biases in times between user actions (e.g., click dwell time) using ideas exploited in PGM-based and neural click models.

Generating responses — 45 minutes (MD, TK).

15 mins Machine reading/question answering – How is the sequence-to-sequence paradigm applied in neural QA systems.

15 mins Conversational IR/dialogue systems – Unlike QA systems, conversational systems should maintain a state of a session.

15 mins General chatbots – Chatbots bring their own set of challenges. How to stay consistent throughout the course of a conversation? How to maintain a persona?

Outlook — 30 minutes (all).

15 mins Recent advances

15 mins Open research questions, current challenges

Wrap up — 15 minutes (TK).

15 mins Overview of material presented and conclusion

4 TYPE OF SUPPORT MATERIALS TO BE SUPPLIED TO ATTENDEES

Slides Slides will be made publicly available on http://nn4ir.com.

Bibliography An annotated compilation of references will list all work discussed in the tutorial and should provide a good basis for further study.

Code Apart from the various open source neural toolkits (Tensorflow, Theano, Torch) many of the methods presented come with implementations released under an open source license. These will be discussed as part of the presentation of the models and algorithms. We provide a list pointers to available code bases.

REFERENCES
