STEVENS INSTITUTE OF TECHNOLOGY

Deep Learning for Recommendation

Rensheng Wang,
https://sit.instructure.com/courses/55957

April 21, 2022

Traditional Matching Models

- The traditional machine learning techniques have been used for conducting query-document matching in search and user-item matching in a recommendation.
- The methods can be formalized within a more general framework, called by us “learning to match”.
- Besides search and recommendation, it is also applicable to other applications such as paraphrasing, question answering, and natural language dialogue.

Learning to Match

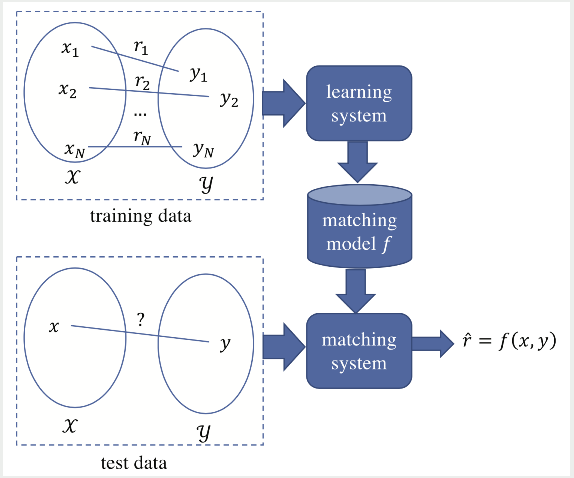
- The “learning to match” problem can be defined as follows.
- Suppose that there are two spaces $\mathcal{X}$ and $\mathcal{Y}$. A class of matching functions $\mathcal{F} = \{f(x, y)\}$ is defined on two objects from the two spaces $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where each function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{R}$ represents the matching degree between the two objects x and y.
- The two objects x and y, and their relationship can be described with a set of features $\Phi(x, y)$. The matching function $f(x, y)$ can be a linear combination of features:

$$f(x, y) = \langle \mathbf{w},\ \Phi(x, y) \rangle,$$

where $\mathbf{w}$ is the parameter vector. It can also be a generalized linear model, a tree model, or a neural network.

Learning of Matching Functions

- Supervised learning can be employed to learn the parameters of the matching function f, as shown in the Figure below.



Learning of Matching Functions

- Supervised learning for matching typically consists of two phases:
  - offline learning
  - online matching
- In offline learning, a set of training instances $D = \{(x_1, y_1, r_1), \cdots, (x_N, y_N, r_N)\}$ is given, where r_i is a Boolean value or real number indicating the matching degree between objects x_i and y_i, and N is the size of training data.
- Learning is conducted to choose a matching function $f \in \mathcal{F}$ that can perform the best in matching. In online matching, given a test instance (a pair of objects) $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the learned matching function f is utilized to predict the matching degree between the object pair denoted as $f(x, y)$.

Learning of Matching Functions

- Similar to other supervised learning problems, we can define the goal of learning to match as minimizing a loss function, which represents how much accuracy the matching function can achieve on the training data as well as the test data.
- More specifically, given the training data D, the learning amounts to solving the following problem:

$$\arg\min_{f \in \mathcal{F}} L(D, f) + \Omega(f)$$

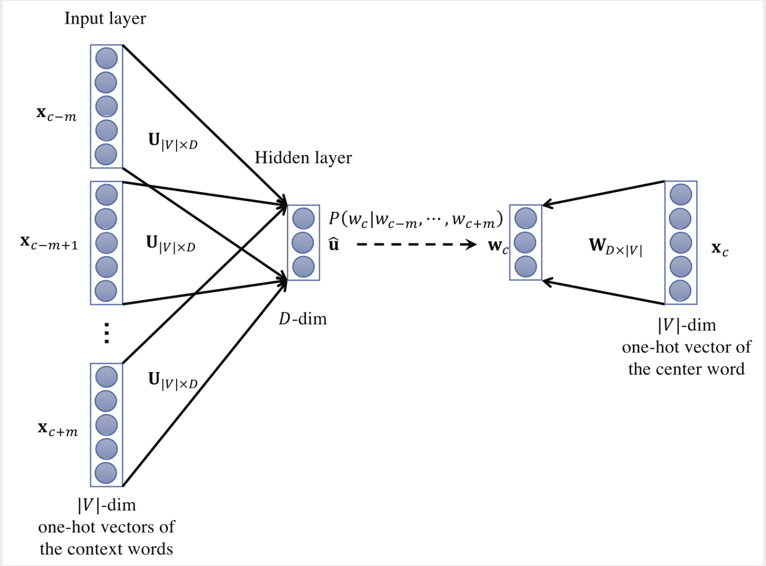
- The objective consists of two parts: the empirical loss $L(D, f)$ measures the overall loss incurred by the matching function f on training data, and the regularizer $\Omega(f)$ prevents overfitting to the training data. $\Omega(f)$ is typically chosen to impose a penalty on the complexity of f. Popular regularizers include $l1$, $l2$, and a mixture of them.

Deep Learning for Matching: Representation Learning

- The strong ability in representation learning is the primary reason for the big success of deep learning, including methods of learning word embeddings and contextualized word representations.
- **Word embedding** is a basic way of representing words in Natural Language Processing (NLP) and Information Retrieval (IR). Embeddings of words are usually created based on the assumption that the meaning of a word can be determined by its context in documents
  - **Word2Vec**: Mikolov et al. (2013) proposed the Word2Vec tool and made word embedding popular. Word2Vec learns embeddings of words from a large corpus using shallow neural networks in an unsupervised manner.
  - There are two specific methods in Word2Vec: Continuous Bag of Words (CBOW) and Skip Gram.

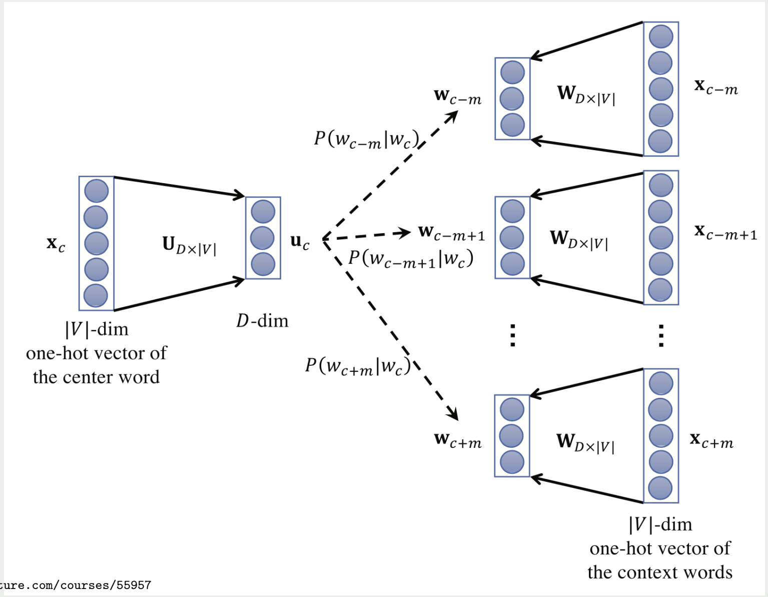
Word2Vec

- Continuous Bag of Words (CBOW)


Input layer
$\mathbf{x}_{c-m}$
$\mathbf{U}_{|V|\times D}$
Hidden layer
$\mathbf{x}_{c-m+1}$
$\mathbf{U}_{|V|\times D}$
$P(w_c|w_{c-m},\cdots,w_{c+m})$
$\hat{\mathbf{u}}$
$\mathbf{w}_c$
$\mathbf{W}_{D\times|V|}$
$\mathbf{x}_c$
D-dim
$|V|$-dim one-hot vector of the center word
$\mathbf{x}_{c+m}$
$\mathbf{U}_{|V|\times D}$
$|V|$-dim one-hot vectors of the context words

Word2Vec

- Skip Gram



Contextualised Word Representations

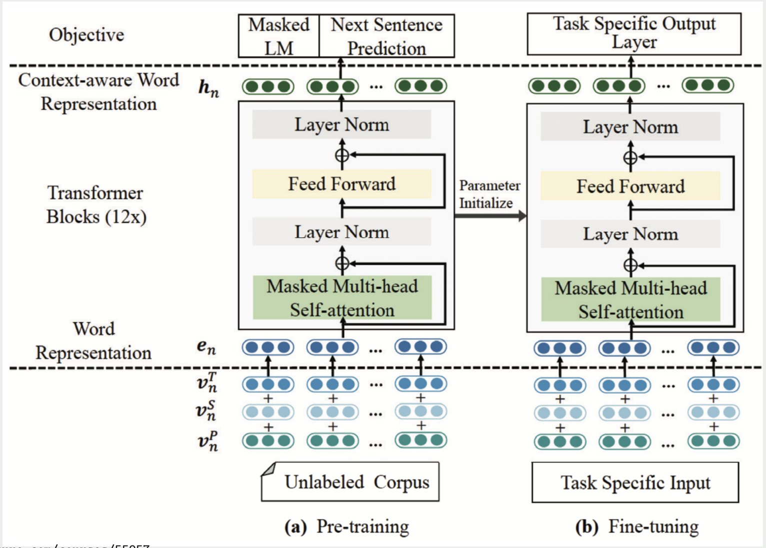
- The classical word embedding models (e.g., Word2Vec and GloVe) have a fundamental shortcoming: they generate and utilize the same embed- dings of the same words in different contexts.
- Therefore, they cannot effectively deal with the context-dependent nature of words.
- Contextu- alized word embeddings aim at capturing lexical semantics in different contexts.
- A number of models have been developed, including ULMFiT (Universal Language Model Fine-tuning), ELMo (Peters et al., 2018), GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), and XLNet (Yang et al., 2019c).

BERT

- Among the models, BERT is the most widely used.
- BERT is a mask language model (a denoising auto-encoder) that aims to reconstruct the original sentences from the corrupted ones.
- That is, in the pre-train phase, the input sentence is corrupted by replacing some original words with "[MASK]". The learning objective, therefore, is to predict the masked words to get the original sentence.

BERT

- BERT training procedure: (a) The pre-training stage and transformer architecture; (b) the fine-tuning stage modifies the pre-trained parameters by task-specific training.

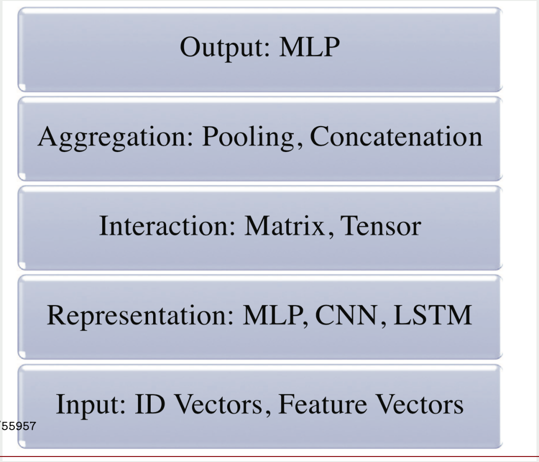


Deep Learning for Matching

- Deep learning for matching, referred to as deep matching, has become the state-of-the-art technology in search and recommendation.
- Compared with the traditional machine learning approaches, the deep learning approaches improve the matching accuracy in three ways:
  - (1) using deep neural networks to construct richer representations for matching of objects (i.e., query, document, user, and item),
  - (2) using deep learning algorithms to construct more powerful functions for matching,
  - (3) learning the representations and matching functions jointly in an end-to-end fashion

General Framework for Deep Matching

- The matching framework takes two matching objects as its input and outputs a numerical value to represent the matching degree.
- The framework has input and output layers at the bottom and the top. Between the input and output layers, there are three consecutive layers. Each layer can be implemented as a neural network or a part of a neural network:

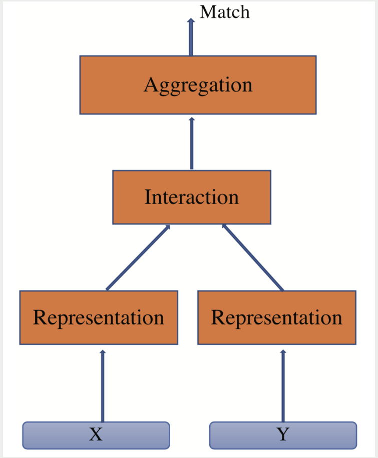

Output: MLP
Aggregation: Pooling, Concatenation
Interaction: Matrix, Tensor
Representation: MLP, CNN, LSTM
Input: ID Vectors, Feature Vectors

General Framework for Deep Matching

- **The input layer** receives the two matching objects which can be word embeddings, ID vectors, or feature vectors.
- **The representation layer** converts the input vectors into the dis- tributed representations. Neural networks such as MLP, CNN, and RNN can be used here, depending on the type and nature of the input.
- **The interaction layer** compares the matching objects (i.e., two dis- tributed representations) and outputs a number of (local or global) matching signals. Matrix and tensor can be used for storing the signals and their locations.
- **The aggregation layer** aggregates the individual matching signals into a high-level matching vector. Operations in deep neural networks such as pooling and concatenation are usually adopted in this layer.
- **The output layer** takes the high-level matching vector and outputs a matching score. Linear model, MLP, Neural Tensor Networks (NTN), or other neural networks can be utilized.

Typical Architectures for Deep Matching

- In the architecture below, the inputs X and Y are two texts in search or two feature vectors in a recommendation. The two inputs are first processed with two neural networks independently, for creating their representations. Then, the architecture calculates the interactions between the two representations and outputs matching signals. Finally, the matching signals are aggregated to form the final matching score.



Designing Principles of Deep Matching

- Two designing principles for the development of deep matching models in search and recommendation:
  - the modular principle
  - the hybrid principle
- The modular principle postulates that a matching model usually consists of multiple modules (functions), and thus the development of such a model should also take a modular approach.
- For example, the representation module can be implemented with CNN, RNN, or MLP, the interaction module can be a matrix or a tensor, and the aggregation module can be a pooling or concatenating operator. Different combinations of the modules result in different matching models.

Designing Principles of Deep Matching

- Two designing principles for the development of deep matching models in search and recommendation:
  - the modular principle
  - the hybrid principle
- The hybrid principle asserts that a combination of dichotomic techniques is helpful in the development of matching models.
- For example, in user-item matching in a recommendation, the first-order, second-order, and higher-order interactions all contribute to the determination of the final matching degree.
- In query-document matching in search, the query and document can be represented with both bag-of-words and sequences of word embeddings.
- Furthermore, in both search and recommendation, the representation and interaction between objects can be combined using a combination of deep and wide neural networks, or nonlinear and linear models.

Matching Based on Representation Learning

- General Framework

  The representation learning methods assume that queries and documents can be represented by low-dimensional and dense vectors.

- There are two key questions: (1) what kind of neural networks to use for creating the representations of query and document, and (2) what kind of function to use for calculating the final matching score based on the representations.

- Formally, given query q in the query space $q \in \mathcal{Q}$ and document d in the document space $d \in \mathcal{D}$, functions $\phi_q : \mathcal{Q} \rightarrow \mathcal{H}$ and $\phi_d : \mathcal{D} \rightarrow \mathcal{H}$ represent mapping from the query space and mapping from the document space to the new space $\mathcal{H}$, respectively. The matching function between q and d is defined as

$$f(q, d) = F(\phi_q(q), \phi_d(d))$$

Representing with Feedforward Neural Networks

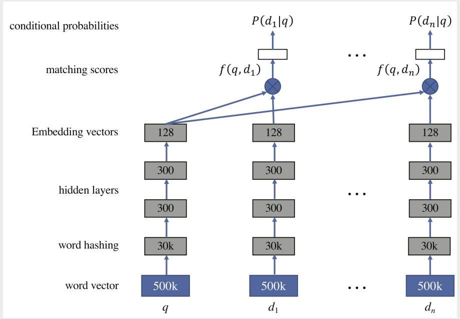
- Feedforward neural networks are the first network architecture used to create semantic representations of queries and documents.
- For example, Huang et al. (2013) propose representing queries and documents with deep neural networks, using a model referred to as Deep Structured Semantic Models (DSSM).
- DSSM first represents query q and its associated documents ds $(d_1, d_2, \cdots, d_n)$ as vectors of terms and takes the vectors as input. To overcome the difficulties resulting from the very large vocabulary size in web search, DSSM maps the term vectors to letter n-gram vectors.
- For example, word "good" is mapped into letter trigrams: ("#go", "goo", "ood", "od#"), where "#" denotes starting and ending marks.

Representing with Feedforward Neural Networks

- In this way, the dimensions of input vectors can be reduced from 500 k to 30 k, because the number of letter n-grams in English is limited. It then maps the letter n-gram vectors into output vectors of lower dimensions through deep neural networks:

$$\mathbf{y}_q = (\text{DNN})(q)$$
$$\mathbf{y}_d = (\text{DNN})(d)$$

where DNN() is the deep neural network used in DSSM, $\mathbf{y}_q$ and $\mathbf{y}_d$ are the output vectors that represent the hidden topics in query q and document d, respectively.



Representing with Feedforward Neural Networks

- Next, DSSM takes the cosine similarity between the output vector of query (denoted as $\mathbf{y}_q$) and the output vector of document (denoted as $\mathbf{y}_d$) as matching score:

$$f(q,d) = \cos(\mathbf{y}_q, \mathbf{y}_d)$$

- DSSM learns the model parameters by Maximum Likelihood Estimation (MLE) on the basis of queries, associated documents, and clicks. Specifically, given query q and a list of documents $D = \{d^+, d_1^-, \cdots, d_k^-\}$, where d^+ is a clicked document and $d_1^-, \cdots, d_k^-$ are unclicked (shown but skipped) documents.

- The objective of learning amounts to maximizing the conditional probabilities of document d^+ given query q:

$$P(d^+|q) = \frac{\exp(\lambda f(q, d^+))}{\sum_{d' \in \mathcal{D}} \exp \lambda f(q, d')}$$

where $\lambda > 0$ is a parameter.