Research Paper Recommender System Based on Deep Text Comprehension

Dongyu Ru[1], **Kun Chen**[1]

[1]IEEE Honer Class – Shanghai Jiao Tong University

Abstract. *Research Paper Recommender System has been exploited over the past few years. Methods based on CBF (Content-based Filtering) , CF (Collaborative Filtering) and GR (Graph-based Recommendations) have been applied to many Research Paper Recommender Systems. And More than half of them used Content-based Filtering. In this report, we proposed a new method of CBF Recommender System for paper recommendations. It uses a deep RNN (Recurrent Neural Network) to dig the information of paper text and construct a matching model to assess the similarity between papers. We also compare the matching model with some traditional baselines with a few experiments to prove our model performs better.*

1. Introduction & Background

Among various kinds of Research Paper Recommender Systems [Beel et al. 2016], CBF [Ferrara et al. 2011] is one of the most widely used and researched recommendation class. The content-based filtering approaches mainly utilized papers that the users had authored, tagged, browsed, or downloaded. TF-IDF was the most frequently applied weighting scheme. In addition to simple terms, n-grams, topics, and citations were utilized to model users' information needs. One central component of CBF is the user modeling process, in which the interests of users are inferred from the items that users interacted with. "Items" are usually textual. "Interaction" is typically established through actions, such as downloading, buying, authoring, or tagging an item. Items are represented by a content model containing the items' features.

However, from our point of view, the feature extractions simply from word-level cannot depict the main content of text information, which leads to relatively lower capacity of content representation. Hence we propose a matching model based on recurrent neural networks. We claim that it has a higher capacity to distinguish and model different kinds of patterns of text content. And therefore, It can achieve a more accurate similarity analysis in a CBF Research Paper Recommender System.

So in this report, based on the traditional CBF Research Paper Recommender System. We replace the original similarity comparison mechanism with a deep neural network which gets higher capacity to dig adequate patterns of text information. Such that more accurate recommendations can be provided with the system.

2. Framework

As we have mentioned above, we use a typical framework of CBF Recommender system as shown in Figure 1. The complete procedure can be described as below:

1. We get tagged papers from users. And the tag interaction is built by the downloading, buying, browsing, marking behaviors the user has manipulated on those papers.

2. We construct the paper corpus from web crawled papers, along with their citation interactions, categories and other meta data.
3. With simple preprocess on papers both from user tagging and papers corpus, we can get a candidate list of papers as potential recommendations and also the user profile. Note that the user profile in our process is actually a set of papers the user interacted with in the history. The weighting scheme on time can be applied here.
4. We take both user profile and candidate papers as input to the DTC model to get the relative similarities between them. Then we can get the recommended papers by similarity rankings.

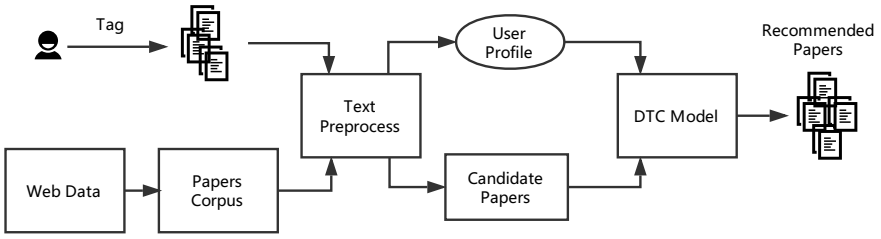

Figure 1. Recommender System Framework

With such framework, we provide user-based personalized recommendations and get a solution for Research Paper Recommender System solution. We saw that the rest parts of the process except for DTC Model are relatively mature and well explored. So we mainly focus on the construction of DTC Model at the following sections.

3. Model (Dongyu Ru)

In this part, we will describe the structure of our DTC (Deep Text Comprehension) Model in detail. Our DTC model is a deep LSTM-based neural network which consists of mainly 7 layers, as shown in Figure 2. It takes as input the words and characters of the paper text. And output a similarity score between the input papers.

1. **Character Embedding Layer [Kim et al. 2016]** This layer maps each word to a vector space using character-level CNN (Convolution Neural Network). Let $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_T\}$ and $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_T\}$ represent the input words of two papers. Characters are embedded into vectors, as 1D inputs to the CNN, whose size is the input channel size of CNN. The outputs of CNN are max-pooled over the entire width to obtain a fixed-size vector for each word.
2. **Word Embedding Layer [Pennington et al. 2014]** This layer maps each word to a high-dimensional vector space. Pretrained word vectors, GloVe, are used to obtain the fixed word embedding of each word. The output of Word Embedding Layer and Char Embedding layer are concatenated together as representation of input text.
3. **Highway Layer [Srivastava et al. 2015]** This layer takes as input the concatenation of two sequences of embedding vectors in word-level. And it performs as

a gate to leak part of original information of input directly to next layer. Let **x** represent the input.

$$\begin{aligned} T(x) &= \sigma(W_T x + b_T) \\ o(x) &= relu(W_o x + b_o) \\ O(x) &= T(x) \cdot o(x) + (1 - T(x)) \cdot x \end{aligned} \tag{1}$$

4. **Contextual Embedding Layer** In this layer, a LSTM(Long Short Term Memory) [Hochreiter and Schmidhuber 1997] Network is applied after the Highway layer output. The output states of LSTM are concatenated and transmitted to the next layer. Till now, feature representation on different granularity has been obtained.

$$y_t = BiLSTM(y_{t-1}, x_t) \tag{2}$$

5. **Attention Flow Layer [Seo et al. 2016]** Here, contextual embedding output of two papers are input to the Attention Flow Layer to get a mutual-aware representation of input papers.
6. **Modeling Layer** The Modeling Layer are constructed by another LSTM layer. The input of modeling layer is attention output stacks. It captures the interaction in the mutual-aware representation of input papers.
7. **Dense Layer** The Dense Layer acts as the output layer of this model, which takes the final state of Modeling Layer as input, use a fully-connected layer and sigmoid function to get score of similarity.

$$score = sigmoid(W_s^T M) \tag{3}$$

4. Baselines (Kun Chen)

In this part, we will describe the two baseline methods (tf-idf and simhash) used in our experiment in detail. tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. simhash is a technique for quickly estimating how similar two sets are. The algorithm is used by the Google Crawler to find near duplicate pages.

1. **tf-idf** The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. The weight of a term that occurs in a document is simply proportional to the term frequency, and the specificity of a term can be quantified as an inverse function of the number of documents in which it occurs. The tf–idf is the product of term frequency and inverse document frequency. In our model, we used the simplest tf scheme, i.e. $tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$, where $n_{i,j}$ is the number of times the word t_i appeared in the document d_j, and $\sum_k n_{k,j}$ is the sum of the times all the words in d_j appeared, i.e. the total number of words in the document. idf is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. $idf_i = log\frac{|D|}{1+|j:t_i \in d_j|}$, where $|D|$ is the number of documents in the corpus, and $|j : t_i \in d_j|$ is the number of documents where the term t_i appears. In order to prevent the division-by-zero problem, we add 1 to the denominator.

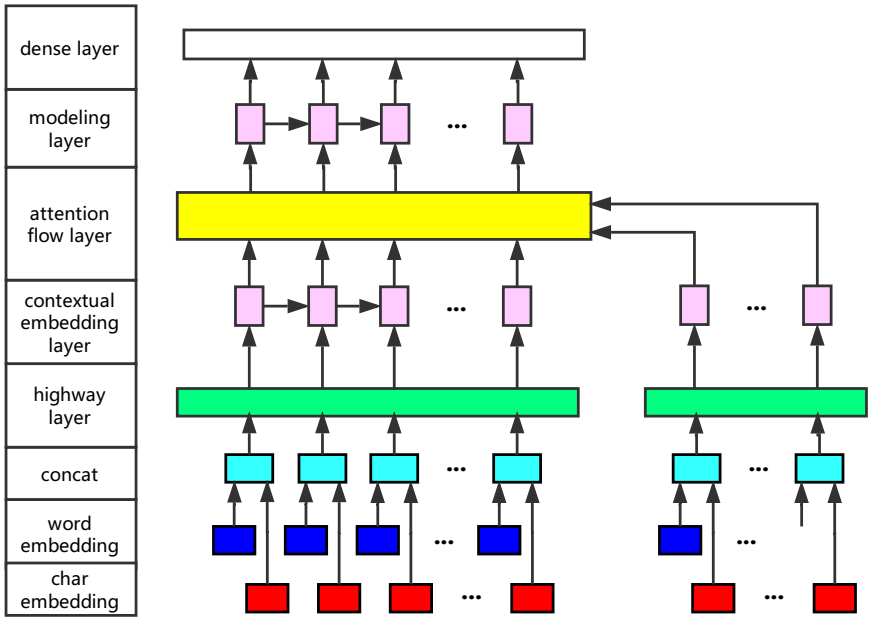

Figure 2. DTC Model

2. **simhash** simhash is a hash function where similar items are hashed to similar hash values, the word ”similar” means the bitwise hamming distance between hash values. The simhash of a phrase is calculated as follow:

   (a) pick a hashsize, for example 64 bits
   (b) let V = [0] * 64 (i.e. 64 zeros)
   (c) break the phrase up into features
   (d) hash each feature using a normal 64-bit hash algorithm
   (e) for each hash,if bit_i of hash is set then add 1 to V[i],if bit_i of hash is not set then take 1 from V[i]
   (f) simhash bit_i is 1 if $V[i] > 0$ and 0 otherwise

   simhash is useful because if the simhash bitwise hamming distance of two phrases is low then their jaccard coefficient is high. in the case that two numbers have a low bitwise hamming distance and the difference in their bits are in the lower order bits, then it turns out that they will end up close to each other if the list is sorted. In order to avoid whether the items differ in the higher or lower order bits we can just do the following 64 times:

   (a) rotate the bits
   (b) sort
   (c) check adjacent

   In our experiment the length of phrase n is much larger than 64, so the entire algorithm will be under $O(n^2)$.

5. Experiments

To prove our model performs better to play as a matching model in Research Paper Recommender System. We collect a dataset to verify the performance of our model and baselines. Restricted by the limited computation power, we randomly selected 1M papers from the dataset for validation. After filtering out bad cases in the dataset. Finally we perform the experiments on a dataset of 200K papers.

30% of the datasets are reserved as test set. And experiments on baselines are directly performed on test set without training. We evaluate our DTC (Deep Text Comprehension) Model with ROC (Receiver Operating Characteristic Curve) as shown in Figure 3 and AUC (Area Under Curve) as shown in Table 1.

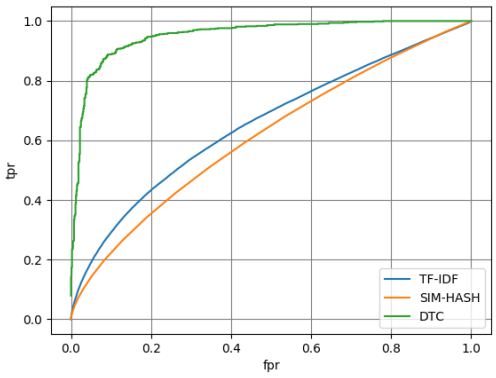

Figure 3. ROC of 3 matching models

Table 1. AUC comparison of matching performance

	TF_IDF	SIM-HASH	DTC
AUC	0.65	0.61	**0.95**

One more thing need to mention is that our model runs slower than those 2 baselines. For a same dataset with 60K data items, TF-IDF runs for 3 mins, while Simhash runs for 3 hours, both on CPU i5-5200U. Our model needs about 10 hours, and extra GPU support needed.

6. Conclusion

In this report, we proposed a Deep Text Comprehension based Recommender System, which replace the original matching model in a CBF Research Paper Recommender System with a Deep Neural Network. And we claim the DTC model has higher capacity to recognize the patterns in text. Experiments indicate that our DTC model performs better than two baselines mentioned in the report. However, the extra running time and computing power is still a problem to be fixed.

References

Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338.

Ferrara, F., Pudota, N., and Tasso, C. (2011). A keyphrase-based paper recommender system. In Agosti, M., Esposito, F., Meghini, C., and Orio, N., editors, *Digital Libraries and Archives*, pages 14–25, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016). Character-aware neural language models. In *AAAI*, pages 2741–2749.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.