2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

ActRec: A Word Embedding-based Approach to Recommend Movie Actors to Match Role Descriptions

Ai-Ni Lee
National Cheng Kung University
Tainan, Taiwan
a0909333247@gmail.com

Kuan-Ying Chen
National Cheng Kung University
Tainan, Taiwan
h24051312@ncku.edu.tw

Cheng-Te Li
National Cheng Kung University
Tainan, Taiwan
chengte@ncku.edu.tw

***Abstract*—In this work, we propose a novel recommendation problem, actor recommendation (ActRec), based on unstructured text data for the movie industry. Given the text description of a role, we generate a ranking list of actors such that the most proper actors for the role-playing can be at top positions. We propose a word embedding-based approach to solve the ActRec problem. In addition, we compile a multi-source data from Wikipedia, Google Search, and PTT online forum. Experimental results show the promising performance of our method, which encourages future effort on ActRec.**

***Index Terms*—Actor Recommendation, User-Generated Data, Movie Analysis, Role description, Semantic Matching**

I. Introduction

Recommender systems (RS) are widely applied in a variety of domains, such as e-commerce, social networking, news recommendation, and search engine. Recent advances on deep learning further improve the performance of recommendation algorithms by better representing users, items, and their context information [14]. With the success of RS based on explicit users-item interactions, nevertheless, it is still an undeveloped land for applying RS to movie industry. In the movie industry, it is highly demanded to recommend proper actors and actresses that match the roles depicted by the directors [7]. We term the task as *Actor Recommendation* (ActRec). Specifically, in actor recommendation, we are given a set of actors, and each actor is associated with a collection of unstructured text descriptions that represent the roles they had ever played and/or their figures and personality. Besides, the directors or producers of a movie can specify the roles in the text form. The task of ActRec is to generate a ranking list of actors that best match the role description. To the best of our knowledge, we are the first to perform such a kind of actor recommendation based on unstructured text data. Although Solaimani et al. [11] have ever identified political actors from news texts, their method relies on hand-coded dictionaries, and does not deal with the semantics of actors.

We develop a word embedding-based approach to solve the proposed actor recommendation task. The basic idea is to map actor names and words that appear in both descriptions of actors and roles into the space embedding space. By doing so, we are allowed to semantically compare actors and any words. Then the recommendation can be cast into retrieving the top-K similar actors by using the query terms in the semantic space. The most challenging part of this task is that currently there is no publicly available datasets on movie actors and roles. Thus we attempt to collect multi-source datasets from Wikipedia, Google Search Engine, and an online forum PTT (the most popular forum in Taiwan) that contains user-generated data on discuss movies and actors.

The contribution of this work is five-fold. First, we propose and define the problem of actor recommendation based on unstructured text description data. Second, we collect and compile a multi-source dataset that contains text descriptions on actors and roles of movies. Third, we propose a word embedding-based approach to deal with the actor recommendation task. Fourth, experimental results show that the proposed method leads to promising performance using all datasets, comparing to several baselines. Fifth, our empirical study provides a list of insights on how to develop a system for description-based actor recommendation.

Related Work. Rolenet [13] analyzes social interactions between roles in a movie based on multimedia content. Hu et al. [3] recommend movie stars based on the historical movie genres and styles. However, their recommendations cannot be tailored to match the role semantics/descriptions. Jakob et al. [4] consider the roles of leading actors/actresses and user reviews for movie recommendation. Chen et al. [1] point out that the recommendation of movies should match the personality of users and roles. Although these studies discuss the roles of actors/actresses are correlated with movie recommendation, none of them exploit the role descriptions for actor recommendation. While Wallace et al. [12] emphasize the selection of proper actors/actresses affects the success of films, our work delivers the first attempt to perform actor recommendation based on role descriptions.

II. Problem Statement

Let A denote the set of actors. Each actor $a \in A$ is associated with a collection of text descriptions, denoted by T_a. The text description of a role r is denoted by T_r. It is

IEEE/ACM ASONAM 2020, December 7-10, 2020
978-1-7281-1056-1/20/$31.00 © 2020 IEEE

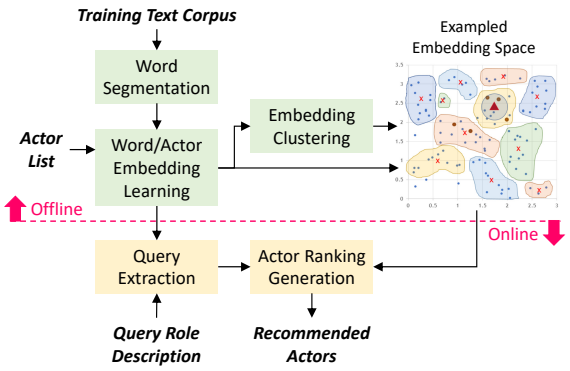

Fig. 1. Flowchart of ActRec model.

worthwhile to mention that T_r can be a subset of T_a if actor a had ever played role r. In other words, T_r consists of two parts. One is the actor itself, i.e., multi-source descriptions on actors. The other is the descriptions of roles that the actors had ever played.

Given the text description $t_{r^\star}$ of a new role $r^\star$, the task of actor recommendation aims at generating a top-k ranking list L of actors from all actors A such that the most *proper* actor $a^\star$ can be ranked at top positions in L. That said, we aim to accurately find which actor will play role $r^\star$ in the future.

III. The Proposed ActRec Method

Our method consists of five phases: (1) preprocessing and word segmentation, (2) word/actor embedding learning, (3) embedding clustering, (4) query extraction, and (5) ranking generation.

We present the flowchart of the proposed ActRec model in Figure 1, in which the upper part is the offline module and the bottom part is the online module. Given the training text corpus and the list of actors as the input, the offline module is to learn the embeddings of words and actors. We first perform word segmentation to obtain the set of words. The word embedding learning technique is used to generate the embedding of extracted words and given actors. That said, words and actors are projected into the same embedding space so that we are allowed to search for relevant actors based on textual terms. To provide effective actor recommendation based on similarity search in the embedding space, we perform embedding clustering such that those words and actors sharing similar contexts in the text corpus are grouped together. In the online module, given the role description, we first extract query terms that are not only representative but also have ever appeared in the embedding space. Last, by similarity search in the embedding space, the actor ranking generation method is devised to return top-recommended actors.

A. Preprocessing and Word Segmentation

Given the original corpus of text descriptions, we first remove all of the stop words to eliminate noises. Then we perform word segmentation to obtain the set of words for recommendation. Since we target at Chinese movies, i.e., text descriptions are Chinese, we exploit the state-of-the-art neural word segmentation technique, ckiptagger[1] [6], to generate the set of words. Note that we follow the hyperparameters suggested by the original paper of ckiptagger to have segmented words.

B. Word/Actor Embedding Learning

Another input of our model is the list of actors. To learn the representations of actors, we require the actors to appear in the text corpus. That said, we filter out those actors that are not contained by the text corpus. To map all of the words and actor names into the same embedding space, we leverage the technique of word embeddings. Glove [9] is used to generate the embeddings of all terms. By doing so, actors and normal terms (e.g., named entities, adverbs, and adjectives) can be compared, and thus we are allowed to find actors based on its surrounding terms in the role description. Note that although Glove is used here, it can be replaced by any recent word pre-training model such as BERT [2].

C. Embedding Clustering

Since some words can be correlated with each other, we perform k-means clustering [5] to group those close with one another. Embedding clustering can benefit the actor ranking generation. It is because the query terms can scatter in the embedding space. If we simply average the embedding vectors of query terms into a center vector, and find the closest actors to the center one, unexpected actors can be reported. For example, there could be a few irrelevant words in the role description. Averaging the embedding vectors of these words can mislead the semantic representation of the input role description. Therefore, we cluster words by their embeddings, and use the clusterings can help filter out irrelevant query terms. The centroid of each cluster is obtained. We will describe how cluster centroids can be used to avoid reporting irrelevant actors, and to find the most representative actors in the section of actor ranking generation. We will also show how the number of clusters affects the performance in the experiments.

D. Query Extraction

Given the text description of a role (provided by movie directors), we extract the corresponding representative query terms. The TF-IDF measure is used to score all words in the description. Top-K high-scored words are regarded as query terms. In the experiments, we will report how K affects the recommendation performance. Note that the scoring of TF-IDF can be replaced by state-of-the-art keyword extraction methods, such as deep keyphrase generation [8].

E. Actor Ranking Generation

The generation of actor ranking consists of two steps. The first step is to find the word/actor cluster that delivers similar semantics as the input role description (query terms). The second step is to produce the ranking list of actors by a

[1]https://github.com/ckiplab/ckiptagger

scoring measure that estimates how an actor matches the role description.

First, to find the representative cluster, we use query terms to vote for clusters. Each query term can vote for a cluster based on the similarity between query term embedding and cluster centroid embedding. The cluster with the highest sum of similarity scores is considered as the representative one. We will recommend actors from the representative cluster.

Second, we generate the ranking list of recommended actors based on two criteria. One is the popularity (termed **pop**) of an actor, i.e., the number of description appearances of an actor in the text corpus. To boost the visibility of a movie and ensure the quality of role playing, it is usually to consider popularity to select actors. The other is the similarity (termed **sim**) between an actor and terms in a word cluster. We first find the cluster $c^{\star}$ whose center vector is closest to the center vector of query terms. Then the embedding similarity is used to rank actors in cluster $c^{\star}$. We can multiply popularity with similarity (termed **popsim**) to be the final ranking criterion.

IV. Experiments

We conduct experiments to answer four questions. (a) Can our method accurately find the most proper actors for a given new role? (b) What is the performance of every dataset in actor recommendation? (c) How do different hyperparameters affect the recommendation performance? (d) How does the popularity of actors influence the recommendation?

A. Data and Settings

Multi-source Dataset. We first find all Traditional-Chinese movies in the past 10 years based on Wikipedia, and find the leading actors and actresses for every movie. There are 973 movies, 1,946 roles, and 2,574 actors. From Wikipedia, we also collect the text descriptions of roles of leading actors and actresses. We retrieve the text descriptions of actors from three sources. (1) Google Search Engine: by using the actor name as the search query, we collect top-20 search-resulting snippets in the first 10 pages of search results. (2) Wikipedia: we collect the text descriptions of each actor. (3) PTT[2]: we first collect all 27,879 articles in PTT Movie Board [3], and retrieve sentences that actors appear as their text descriptions.

Evaluation Settings. Movies from 2010-2018 are used to learn the embeddings of all words and actors, and role descriptions of 2019 movies are used for testing. We use *Mean Average Precision* (MAP) as the evaluation metric. We set the dimensionality of embeddings to be 128. The number of clusters is set as 30 by default. Lists of top-10 actors are generated. The competing methods are different combinations of word vector with clustering (cwv) or simply averaging (wv), and utilizing popularity with similarity (popsim) or not (pop), i.e., **wv-pop**, **wv-popsim**, **cwv-pop**, and **cwv-popsim**. Note that wv indicates simply using the averaged embedding vector of all terms in the role description to represent the query. We consider four sets of actor text descriptions, including Wiki, Google Search, PTT, and ALL of them.

[2]https://term.ptt.cc/

[3]https://www.ptt.cc/bbs/movie/index.html

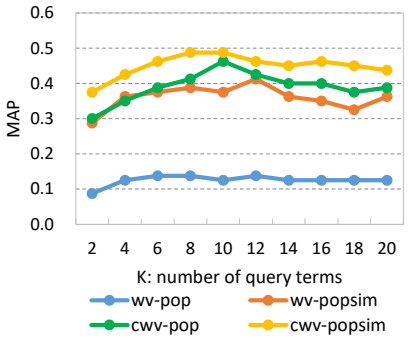

Fig. 2. MAP on different methods using ALL data.

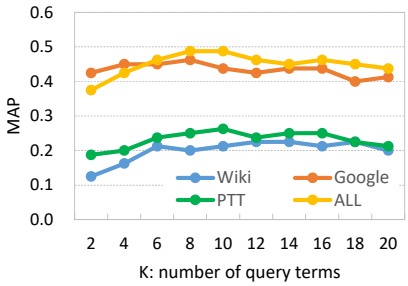

Fig. 3. MAP on different datasets using cwv-popsim.

B. Evaluation Results

Main Results. The results are shown in Figure 2 and Figure 3. From Figure 2, which varies the number of query terms by using their TF-IDF scores (terms with high TF-IDF scores are first considered), it can be found that the proposed method word embeddings with clustering under the ranking criterion popularity-similarity (i.e., cwv-popsim) leads to the best performance. When considering a proper number of query terms (e.g., $K = 8, 10$), we have better MAP results. We think that too many query terms can bring words that are not so relevant to the role description, and fewer words can fully depict the semantics of a movie role.

On the other hand, in Figure 3, which also varies the number of query terms, the ALL data generates the most satisfying results. We think the possible reason is that different datasets are able to capture semantics from various aspects. Collective opinions are captured from users in PTT. The official descriptions about the personality and impression of an actor/actress can be found in his/her Wikipedia page. Google search results provide the news about the actor/actress. ALL that combines semantics from multiple aspects can better and fuller depict the actor/actress.

More Analyses. We first present how different word embedding dimensions and various number of clusters affect the recommendation performance. By changing the embedding dimension (32, 64, 128, 256), the MAP scores of cwv-pop and cwv-popsim are shown in Figure 4 (left). It can be found that we have better performance when the embedding dimension is 128. Higher or lower dimensions can hurt the recommendation.

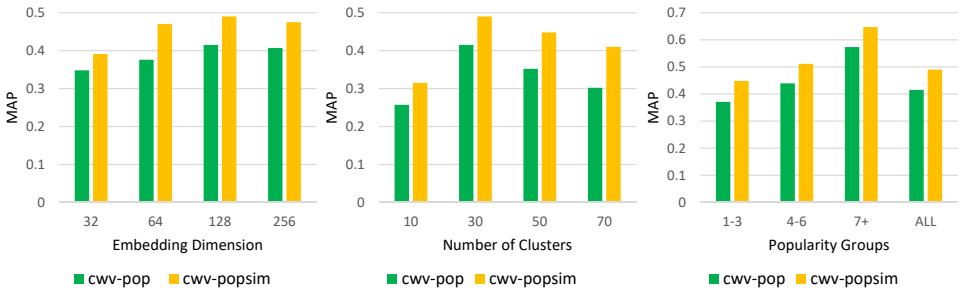

Fig. 4. Experimental results on hyperparameter analysis (the left is the embedding dimension, and the middle is the number of clusters), and the effect of actor popularity groups (right) between cwv-pop and cwv-popsim methods.

The results of changing cluster numbers (10, 30, 50, 70) are exhibited in Figure 4 (middle). We obtain higher MAP scores when the number of clusters is 30. Although the empirical results show that the performance is a bit sensitive to the embedding dimension and the number of clusters, we think the choice of such hyperparameters should rely on the text corpus and how different types of movies and roles are considered. They should be adjusted and tailored to the collected datasets.

We also demonstrate how the popularity of actor/actress influences the recommendation performance. We group the input role descriptions based on the popularity of their corresponding ground-truth actors/actresses, in which the popularity is defined as the number of played movies. Four popularity groups, including 1-3, 4-6, 7+, and ALL, are considered. The results in MAP scores are presented in Figure 4 (right). We can clearly find that role description groups with popular actors/actresses lead to better performance, especially on the "7+" group. We think such results bring an essential implication. Popular actors/actresses are discussed more frequently in social media (PTT), have more news articles, and are described using more sentences. They have much richer training texts. Therefore, we can generate more effective word embeddings to depict these actors/actresses. In contrast, the unpopular actors have fewer text descriptions, and thus their recommendation leads to worse performance.

In short, we arrange insights found in the experimental studies. (1) To have a more accurate recommendation of movie actor/actress, we need to collect more text descriptions for learning better embeddings. (2) The text descriptions should be collected from diverse sources (e.g., news, social media, and Wikipedia). (3) An effective mechanism to select representative query terms from the input role description is also crucial. (4) Both actor popularity and embedding similarity are equally important in recommending actors. (5) It is also essential to filter out irrelevant actors, and our embedding clustering and voting is a potential strategy.

V. Conclusions

This paper presents a novel actor recommendation problem based on unstructured text data for the movie industry. We collect a multi-source dataset, and propose a word embedding-based approach to deal with the task. Promising experimental results encourage future effort on improving actor recommendation. The empirical study also provides a list of insights on description-based actor recommendation. Ongoing work is to exploit the semantic match technique [10] to directly learn the matching function between descriptions of actors and roles.

References

[1] L. Chen, W. Wu, and L. He, *Personality and Recommendation Diversity*, 2016, pp. 201–225.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[3] Y. Hu, Z. Wang, W. Wu, J. Guo, and M. Zhang, "Recommendation for movies and stars using yago and imdb," in *2010 12th International Asia-Pacific Web Conference*, 2010, pp. 123–129.

[4] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych, "Beyond the stars: Exploiting free-text user reviews to improve the accuracy of movie recommendations," in *Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, ser. TSA '09, 2009, pp. 57–64.

[5] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[6] P.-H. Li, T.-J. Fu, and W.-Y. Ma, "Remedying bilstm-cnn deficiency in modeling cross-context for ner," in *Proceedings of AAAI International Conference on Artificial Intelligence (AAAI)*, 2020.

[7] A. Liu, Y. Liu, and T. Mazumdar, "Star power in the eye of the beholder: A study of the influence of stars in the movie industry," *Marketing Letters*, vol. 25, no. 4, pp. 385–396, 2014.

[8] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017, pp. 582–592.

[9] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[10] J. Rao, L. Liu, Y. Tay, W. Yang, P. Shi, and J. Lin, "Bridging the gap between relevance matching and semantic matching for short text similarity modeling," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5369–5380.

[11] M. Solaimani, S. Salam, L. Khan, P. T. Brandt, and V. D'Orazio, "Apart: Automatic political actor recommendation in real-time," in *Social, Cultural, and Behavioral Modeling*, 2017, pp. 342–348.

[12] W. T. Wallace, A. Seigerman, and M. B. Holbrook, "The role of actors and actresses in the success of films: How much is a movie star worth?" *Journal of Cultural Economics*, vol. 17, no. 1, pp. 1–27, 1993.

[13] C. Weng, W. Chu, and J. Wu, "Rolenet: Movie analysis from the perspective of social networks," *IEEE Transactions on Multimedia*, vol. 11, no. 2, pp. 256–271, 2009.

[14] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, 2019.