Enriched Network Embeddings for News Recommendation

Janu Verma
Hike Messenger
New Delhi, Delhi, India
j.verma5@gmail.com

ABSTRACT

News aggregators collects content from various sources and presents them in one website or mobile application for easy access. A key challenge for the news applications is to help users discover relevant articles. Both the user experience and the key metrics depend on the high-quality personalized recommendations. However, building a news recommendation presents a set of challenges due the large number of articles being published every hour, the surge and decline in the popularity of news, and critical nature of recency etc. In this paper, we present a graph-based news recommendation model which is deployed on a real-world news application. Our system is a hybrid of collaborative-filtering and the content-based filtering. We enrich the user-article interaction graph by adding new nodes corresponding to the named entities extracted from the contents of the articles. The random walk based graph embeddings are used to learn latent representation for users, articles and named entities in the same space. We evaluate the learned embeddings via a multi-class classification of news articles into high-level categories. We propose a recommendation system based on the binary classification problem which takes as input a combination of the user, item and entity embeddings and computes the probability of the user clicking on the article. We perform experiments to show the superiority of our model to the previous system.

KEYWORDS

News recommendation, network embeddings, NLP, Named Entities, Topic Modeling, Collaborative Filtering

1 INTRODUCTION

News reading on the mobile devices has become very popular in recent years owing to the availability of tremendous amount of data coming from millions of sources across the world. As per a 2017 survey by Pew Research [8], about 85% of US adults read news on a mobile device. News applications like Google News, Bing News, Flipboard, Pocket etc. collect news from various sources and provide readers with news from around the world in an aggregated user-interface. The sheer volume of available articles can be overwhelming to the readers. A key problem that news aggregation applications struggle with is to help users discover articles that are most interesting to them.

Recommendation system aims to capture users preferences and interests so that relevant information is shown to them. Domains like online shopping (e.g. Amazon), movies (e.g. Netflix), music (e.g. Pandora) have seen great success of recommendation systems. Two popular approaches of recommendation systems are collaborative filtering and content-based filtering which form the basis for major recommendation systems. In collaborative-filtering, news articles are recommended based on the reading history of users with similar preferences. This is a very popular method which has the advantage of being domain free. A major drawback of collaborative-filtering is its inability to handle new content i.e. a breaking news that has not yet seen much traction, this is called the *item cold-start problem*.

A solution of the problem of fresh content would be to build user profiles comprising of genuine interests and use them to make news recommendations. This is called content-filtering where the contents of the articles are analyzed to extract topics of user's interest. If a user's past reading preferences are known, new articles can be recommended based on their similarity to the previously read. Content-based filtering requires sufficient reading history of users to be able to build a strong profile of user's interests. For users with little history, i.e. *user cold-start*, this becomes problematic. Collaborative-filtering relies on the fact that there are always users who read some news, and these users may serve as a basis to help to predict the interests of the long-tail users. Thus, collaborative-filtering and content-based filtering are complementary to each other.

The user preferences are not straightforward—a user might want to read an article even if she is not interested in the topic but finds the particular story relevant. For example, wanting to read news about World Cup even if no general interest in Sports. This requires a carefully designed content-filtering model at a proper level of granularity. Furthermore, not all users are equal to each other, and the collaborative filtering method may not account for the individual variability between users. Highly read topics are recommended to most of the users, even if some of the them have no interest in these topics. For example, entertainment and lifestyle articles are most popular and they get reflected in the recommendations for a lot of users via other seemingly similar users.

Thus, news recommendation presents challenges that do not exist in other domains. For example, the recency and the popularity of news articles can change drastically with time. Another compounding factor is the influx of a large number of new articles every hour.

Present Work: In this work, we propose a graph-based news recommendation system which combines the collaborative-filtering and content information. Our model is based on graph embeddings [3], [5] where we map nodes of the graph to vectors in a low-dimensional space such that the structural attributes of the graph translate to the geometrical properties in the embedding space. The user-item interaction data can be defined as a bipartite graph with user nodes and item nodes. In collaborative filtering, the adjacency matrix of this bipartite graph is used to learn similarity of users and items [10]. In the current work, we enrich the user-item graph by adding new nodes corresponding to the named entities - person, place, organization, event - extracted from the contents of the articles. For instance, if an article is about 2016 Presidential elections, we might add entity nodes like - Donald Trump, 2016 Elections, Republican Party etc. Also added are the edges from the item nodes to the entity nodes. The enriched graph has three types of nodes - user, item, entity, and 2 types of edges - user-item and item-entity.

To learn node embeddings, truncated random walks (biased or unbiased) of fixed length are generated emanating from each node [5]. These random walk sequences can be thought of as sentences in an artificial language. Using the Skip-gram [7] model where words in a corpus of sentences in a natural language can be mapped to a low-dimensional space, we obtain dense representations for nodes. The embeddings aim to capture contextual similarity of the nodes i.e. nodes co-occurring on a fixed window on a random walk are mapped to nearby points. In collaborative-filtering nodes which co-occur in the adjacency list (direct connections) of a node are mapped to nearby points [10]. Thus, graph embeddings provide an extension of the collaborative-filtering and has been shown to perform better.

Using graph embedding, we map all nodes—users, items and entities—to the same space. In this space, we can compute similarity of a user to an item and also the user-entity similarity. This allows us to suggest news articles based on a user's affinity towards the entities which are at a much lower level than topical affinity in content-based filtering. This equips us to handle the situation where e.g. a consumer is interested in reading the news about elections even if she is not generally interested in politics. The entities are also more interpretable than abstract topics.

We provide an evaluation of the learned representation by studying its efficacy in multi-class classification of the article nodes into 8 pre-defined high-level categories e.g. Politics, Entertainment, Sports etc. A simple linear classifier trained only on the node embeddings of 60% of the article nodes (without using any content information explicitly) labeled with their respective categories, and evaluated on the remaining 40% provides 0.901 AUC. We also cluster embeddings of the entity nodes and qualitatively evaluate the results. Finally, we evaluate the system for article recommendation by computing *Precision@k* for k values ranging from 1 to 5.

Concretely, we make following contributions:

- Provide a news recommendation system based on graph embeddings that is a hybrid of collaborative and content-based filtering.
- Learn embeddings for the enriched user-item graph that contains entity nodes capturing contents of the article in addition to the user and item nodes.
- Evaluate the learned embeddings via multi-class article classification.
- Build and evaluate the binary classification model for recommendation.
- Study the efficacy of our method for cold-start problem.

The remainder of the article is organized as follows: In Section 2, we provide a discussion of the related work on recommendation systems and network embeddings. Next, we explain graph embeddings for the bipartite and the enriched graphs and their utilization in the recommendation model in Section 3. Section 4 provides the analysis and evaluation of the proposed model. Finally, we conclude in Section 5.

2 RELATED WORK

In this section, we discuss some of the related work on news recommendation systems.

Collaborative Filtering: Collaborative filtering [10] recommends articles which were clicked by users with similar reading history. Collaborative filtering has been applied to personalized news reading applications, such as GroupLens [11] and in the initial version of Google News recommendation [9]. There are two types of collaborative-filtering [10] : em neighborhood model and *latent factor model*. In neighborhood model, the click history of users is used to compute the 'neighborhood' of a user and then articles are recommended to users from the articles clicked on by their neighbors. In the latent factor model, a latent representation of both users and articles in the same space, usually via a matrix factorization of the user-article interaction matrix. The latent representation can be interpreted as describing an article or a user in a 'concept' space which captures the factors e.g. topic of the article. The collaborative-filtering is employed when there is scarce click history available for users.

Content-based Filtering: A Content-based recommendation system tries to recommend items similar to those a given user has liked in the past. Thus, it requires a notion of similarity of articles. There has been a lot of work in NLP to compute similarity between text documents e.g. tf-idf, word embeddings [7] and doc2vec [12]. This relies on sufficient click history to be able to build a genuine profile of user's interests. The content-based filtering has been applied to personalized news recommendations e.g. news reading on devices ([13] , [15]) and web-based news aggregation services [14].

Hybrid Collaborative and Content-based Filtering: Hybrid model which combine both collaborative-filtering and content-based filtering are more stable against the problems of any one of the approaches. This is accomplished by using both the user similarity based on historical information and the content similarity to make recommendation. Hybrid methods have been seen applications in news recommendation [16] and [17].

Graph Embeddings: Our approach presented in this paper is based on graph embeddings. Instead of working with global summary attributes of the graph, there has been a lot of work recently to find a representation of the nodes that incorporates the local structural information [5], [3]. The idea is to learn a mapping from the graph to a low-dimensional real vector space such that the structural attributes of the graph translate to the geometrical properties in the embedding space. Random walks of pre-decided length are generated starting at each node in the network to produce "sentences" of nodes, similar to sentences of words in a natural language. The Skip-gram algorithm devised by Mikolov et al. [7] is used to obtain node embedding from the random walks, which are expected to capture the contextual properties of the network nodes : the nodes that occur in same context have similar vector embedding. For a survey on graph embeddings, see Cui et al. [6]. Recently, random walks on graph and embeddings have been used in recommendation tasks e.g. Pixie [18], GraphSage [19] etc.

3 PROPOSED METHOD

In this section, we describe our model for news recommendation which uses graph embeddings as feature learning. News aggregation applications gather data from various sources and present the results as a list to the users. The recommendation system attempts to rank the list of articles according to a user's preference. Mathematically, the articles are ranked based on the probability of the user clicking on them. We model this as a binary classification problem which takes a user and an article as input and computes the probability of click i.e. $Prob(click = 1|user, item)$. The input features for this model are a combination of the dense representation of the user and the article which we learn through graph embeddings.

During their activity on the application, users interact with many articles. More formally, the user-article interaction can be organized as bipartite graph $G = (U, A, E)$, where U denotes the set of users and A denotes the set of articles. The set $V = U \cup A$ is the set of nodes of G. There is an edge $e \in E$ between a user $u \in U$ and an article $a \in A$ if the u clicked on the snippet for a to read the story. The set of all nodes connected to a node n, called the adjacency list of n is denoted by $E(n)$.

Graph embedding for the bipartite graph: Graph representation learning techniques such as DeepWalk [5] and Node2vec [3] use random walks to embed a graph onto a low-dimensional space which maps each node to a dense vector. Following DeepWalk [5], we generate short truncated random walks starting from each node. The random walks are generated in a completely uniform and unbiased fashion, each adjacent node has equal probability to be picked. The random walks produce sequences of nodes of pre-decided length. For the bipartite graph, the random walks are generated by repeating the operations - 1) Given the current node n which is initialized at the starting node of the random walk, get its adjacency list $E(n)$. 2) Sample an edge from $E(n)$ which links n to the node m. 3) Thee current node is updated to m and the steps repeat. The procedure is described in Algorithm 1. The random walks on the bipartite graph have paths of the form

$$User \rightarrow Article \rightarrow User$$

Algorithm 1 Bipartite Graph Embedding

Input: Bipartite graph $G = (U, A, E)$, walk length l, walks per node r, embedding dimension d, context window size w.

```
1:  Initialize $walks$ to $Empty$.
2:  for $i = 0$ to $i = r$ do
3:    for $n \in V$ do
4:    Initialize $curr\_walk$ to $[n]$.
5:      while $step = l$ to $l$ do
6:        $curr\_step = curr\_walk[-1]$
7:        $adj\_list = E(curr\_step)$
8:        $next\_step = RandomSample(adj\_list)$
9:        Append $next\_step$ to $curr\_walk$
10:   Append $curr\_walk$ to $walks$
11: $SkipGram(walks, w, d)$
```

Output: Embeddings of every node $n \in V$, $v_n \in \mathbb{R}^d$

The random walks thus generated are sequences of the nodes, which can be thought as 'sentences' in an artificial language. The *SkipGram* [7] model, developed for word embeddings, measures the probability of two words to co-occur within a fixed window on a sentence. It uses a fixed size window around every word to extract *context* and *non-context* words for the word under consideration. The model employs a single hidden layer neural network to learn a mapping from the word to its context word. In graph embeddings, the skip-gram model [7] computes the probability of two nodes to co-occur within a fixed window on a random walk.

This procedure is an extension of the latent factor models [10] of collaborative-filtering where matrix factorization is used to obtain dense representations of users and items in the same space. Graph embeddings have been shown to perform better than the matrix factorization for various tasks e.g. classification, clustering, link prediction etc. Mathematically, graph embeddings extend the contextual similarity from immediate neighborhood to truncated random walks. Graph embedding for the user-item bipartite graph, though effective, suffers from the problem of cold start i.e. it's unable to handle new content which was not the part of the graph.

Content-based filtering has been an alternative to collaborative-filtering for handling the problem of fresh, unseen items. We merge collaborative and content filtering in a natural way in our setting by enriching the bipartite graph G to a new graph $G' = (U, A, C, E, E')$ where C denotes set of content nodes and $E')$ is the set of edges between the article and the content nodes. We use the named entities of the form place, person, organization as the content nodes. The named entities are at a finer level than topic modeling appraoch [20] and are more constrained than the tf-idf based keyword filtering [21]. The set of topics and keywords is not exhaustive, new articles can introduce topics and keywords, thus the models for extracting topics or keywords need to be retrained. Named Entity Recognition (NER), in turn, is article agnostic. It can extract entities in new articles without being retrained explicitly. In fact, there are off-shelf tools e.g. Spacy [22] to extract names entities on any article. Thus, NER is cheaper to achieve than topic modeling and keyword extraction.

Graph embedding for the enriched graph: The enriched graph G' contains user-user interaction via their shared interests in articles as well as the article-article interaction via the co-occurrence of entities in them. In the bipartite setting, the random walks transition between user and article nodes. While, in the enriched graph, the random walks are more complex. The random walker at the article node can either jump to the user node or to the entity node. The paths in the random walks can be of following types:

- $User \rightarrow Article \rightarrow User$
- $User \rightarrow Article \rightarrow Entity \rightarrow Article \rightarrow User$
- $Article \rightarrow User \rightarrow Article$
- $Article \rightarrow Entity \rightarrow Article \rightarrow User$
- $Article \rightarrow Entity \rightarrow Article \rightarrow Entity$

This provides a lot more possibilities then the $Article \rightarrow User$ and $User \rightarrow Article$ choices in the bipartite. Thus, the embeddings learned on these random walks are able to capture relationships between different types of nodes due to different types of ways they can be connected.

The enriched graph G' is heterogeneous with two types of edges i.e. user-article and article-entity. The graph embedding methods like DeepWalk [5] are restricted to homogeneous graphs, and may not be directly applicable to heterogeneous networks. One obvious problem is that by having unbiased random walks, we assign equal probability to each edge type. This increases the probability of the random walk going through an edge-type which has multiple edges from the current node. For any article node in the enriched graph, there are far more edges to the user nodes than to the entity nodes. Thus the random walk has higher chances of going to the user nodes, and in some cases completely avoid entity edges. This stems from an imbalance at the global level, we have a tens of thousands of entities, where as there are millions of users. We provide a simple way to resolve the problem of random walks being biased by the dominant edge type. The random walk is generated in two steps: (i) an edge type is chosen randomly from all possible edge types, (ii) an edge is randomly chosen from all edges of the selected edge type. This amounts to biasing the random walks uniformly with equal weights for each edge type. The procedure for learning representations of the nodes for the enriched graph is described in the Algorithm 2 and Algorithm 3.

In reality, different edge types contribute differently to the random walks and would have unequal weights. However, we do not have an intuitive way to obtain these weights. Some work has been done in this direction e.g. Metapath2vec metapath2vec, heterogeneous edge embeddings [23] etc.

Algorithm 2 Enriched Graph Embedding

Input: Enriched graph $G = (U, A, C, E, E')$, walk length l, walks per node r, embedding dimension d, context window size w.

```
 1: Initialize $walks$ to $Empty$.
 2: for $i = 0$ to $i = r$ do
 3:   for $n \in V$ do
 4:   Initialize $curr\_walk$ to $[n]$.
 5:     while $step = l$ to $l$ do
 6:       $curr\_step = curr\_walk[-1]$
 7:       $next\_step = NextNode(G, curr\_step)$
 8:       Append $next\_step$ to $curr\_walk$
 9:   Append $curr\_walk$ to $walks$
10: $SkipGram(walks, w, d)$
```

Output: Embeddings of every node $n \in V$, $v_n \in \mathbb{R}^d$

Algorithm 3 NextNode

Input: Enriched graph $G = (U, A, C, E, E')$, $curr\_node$

```
 1: If $curr\_node \in U$
 2:   $next\_step = RandomSample(E(curr\_node))$
 3: else if $curr\_node \in C$
 4:   $next\_step = RandomSample(E'(curr\_node))$
 5: else if $curr\_node \in A$
 6:   Generate a random number $t$ from $Uniform(0, 1)$.
 7:   If $t > 0.5$
 8:     $next\_step = RandomSample(E'(curr\_node))$
 9:   if $t \leq 0.5$
10:     $next\_step = RandomSample(E(curr\_node))$
```

Output: Next node in the random walk $next\_step$

Recommendation Model: Having learned the embeddings for the users, items and the entities, we now describe the model for recommendation. We define the recommendation problem as a binary classification model which learns the probability of a user clicking on an article. We use logistic regression model on the embeddings learned on the graph. The input *feature vector* to the logistic regression model is the Hadamard product of the *user feature vector* and the *article feature vector*. If both the user and the article were present in the enriched graph i.e. we have an embedding for them, then the user and article feature vectors are their respective graph embeddings. The purpose of a news recommendation system is to recommend new, unseen articles, and we may not have an embedding for the fresh articles. In this case, we use the average of the embeddings of the entities present in the article as the *article feature vector*. Thus the input feature to the logistic regression is in the same space as the graph embedding space.

If $F_u \in \mathbb{R}^d$ is the vector representation of user u and $F_a \in \mathbb{R}^d$ be the vector representation of article a, then the input to the logistic regression model is a vector $x_{(u,a)} = (x_1, x_2, \ldots x_d) \in \mathbb{R}^d$ defined as:

$$(x_{(u,a)})_i = (F_u)_i * (F_a)_i \tag{1}$$

The user feature vector is defined as the user embedding vector V_u learned via graph embeddings. The article feature vector F_a is given by the graph embedding $V_a \in \mathbb{R}^d$ if a has an embedding. Else, if $C(a)$ is the set of entities in the contents of a

$$F_a = \frac{1}{|C(a)|} \sum_{c \in C(a)} V_c \tag{2}$$

The pipeline for recommendation system described above as three components:

- **Feature Learning:** A large part of the user-article interaction data between time t_0 and t_1 is taken to build the enriched graph and then embeddings are learned on this graph. e.g. the user activity on the application between Jan 2016 to Jan 2019.
- **Recommendation Model Training:** The logistic regression model for the computing probability of a user clicking on an article is trained on a different sub-graph which comprises of interaction data between time $t1$ and $t2$ e.g. between Jan 2019 to Jun 2019. A portion of this training data is held out to validate the model.
- **Evaluation and Deployment:** The trained model is then evaluated on the unseen data by considering the predictions of the model on usage between June 2019 to July 2019. The model is then deployed to the application.

An alternate interpretation of the recommendation model is as the link prediction [24] in a user-article bipartite graph selected at a future time i.e. it contains users and new articles.

4 RESULTS AND ANALYSIS

In this section, we provide discussion on evaluation and analysis of the graph embeddings and the recommendation model.

4.1 Experimental Setup

The data for this experiment was taken from a real-world news aggregation mobile application X[1]. The news content on X is presented as a scroll-able list which shows the headline and first couple of lines of the article. The user can read the full article by clicking on the shown snippet for that article. During their activity on the platform, users click on the displayed article snippet if they want to read the full story or they scroll past it. We consider user clicking on an article as an indicator of their interest in the article and use the clicks as the positive data for the recommendation model. If a user scrolls past an article and does not click on it, we use that as the negative example. Thus, our model attempt to increase the click-through rate (CTR) as the metric. For learning the graph embeddings, we create an enriched graph using the activity of 500K users during the period of 3 months, adding 40K articles. There are also 6K entity nodes, which we extracted using the freely available Spacy API.

[1]Name annonymized for the double-blind review.

Table 1: Comparison of Embeddings

Features	Precision	Recall	F-1
word2vec	0.90	0.92	0.91
doc2vec	0.91	0.92	0.91
node2vec	0.88	0.89	0.88

Table 2: Confusion Matrix of the Evaluation

Category	Precision	Recall	F-1
Business	0.83	0.71	0.77
Sports	0.90	0.92	0.91
Entertainment	0.97	0.95	0.96
Tech	0.86	0.83	0.84
Local	0.88	0.95	0.91
World	0.82	0.68	0.75
Lifestyle	0.93	0.85	0.89
Offbeat	0.87	0.62	0.72
Average	0.88	0.89	0.88

To learn embeddings of all the nodes in the enriched graph, we generate 30 random walks of length 100 for every node. The skip-gram model is trained using stochastic gradient (SGD) with a learning rate of 0.01 to minimize the negative sampling loss. Finally, we obtain an embedding of 128-dimension for every node.

4.2 Multi-class Article Classification

We evaluate the feature representations obtained through graph embeddings on a standard supervised learning task - multi-class classification of the news articles. News publishers often add some high level categories to the articles they publish e.g. Sports, Entertainment etc. We train a machine learning model on a the set of the labeled articles using their graph embeddings as input. The task is to predict the labels for the remaining articles. There are 8 news categories in our data - *Business, Entertainment, Sports, Local, Tech, World, Lifestyle, Offbeat.* In this experiment, we use a fraction of the labeled article nodes to train a multinomial logistic regression model with L2 regularization. Without explicitly using any content information, the performance of thee logistic regression model trained on node embeddings (node2vec) is similar to that of a model trained explicitly on content features e.g. word2vec [7] or doc2vec [12]. The comparison is presented in Table 1. The detailed results of the model are described in the confusion matrix in Table 2.

4.3 Recommendation Evaluation

The recommendation model is trained on a set of user-article pairs of positive examples (user clicked) and negative examples (no click) taken from the activity over a period of one month which is different from the data used for building the enriched graph. The logistic regression with L2 regularization is used for binary classification of the pairs as click or no-click. We then evaluate the model to predict whether a

Table 3: Comparison of Recommendation Models

Model	AUC	Precision@5
Hybrid CF and Content	0.72	0.89
Enriched graph embeddings	**0.87**	**0.97**

set of user-article pairs were clicked on not. We measure the performance of the model by computing the Area Under the ROC curve (AUC) and Precision at 5. The model shows a significant improvement over the previously used approach which is a combination of collaborative-filtering and word embeddings (Hybrid CF and Content). One major advantage is that our model is that it naturally blends collaborative-filtering and content-based filtering by putting both the user, item, and the content into the same space.

5 CONCLUSION AND FUTURE WORK

In this work, we proposed a graph-based news recommendation system which is a hybrid of collaborative-filtering and content information-filtering. Our method employs graph embeddings to automatically learn latent representation of users and articles. We extend the user-item bipartite graph to contain names entities from the articles. The entities are expected to capture the user preferences at a finer-level. The embedding methods bring user, items and the entities in the same space. We evaluated the learned latent representations via classification of article nodes into 8 high-level categories. We show that without explicitly using the contents of the article, we achieve results comparable to the NLP based features. We also design and evaluate a recommendation model as a binary classification model for computing the likelihood of the user clicking on the article. This model performs better than the hybrid collaborative-filtering and word embeddings based article similarity model.

Though the model performs satisfactorily, this work has focused on simplicity since the goal of this paper is to show the efficacy of graph embedding methods for content recommendation. The embedding method has two hyperparameters - walk length and number of walks per node - which we chose based on heuristics and did not learn them. There is work using attention mechanism [27] to learn the hyperparameters as an end-to-end system. The enriched network we used has two types of edges. We resolved the heterogeneity by giving equal importance to each edge type. Graph embeddings for heterogeneous networks is an active area of research e.g. metapath2vec [2], heterogeneous edge embeddings [23] etc. The logistic regression model for recommendation was chosen its simplicity, a more suitable approach would be a neural network based model which is either trained on Siamese like loss (two input - user and item embeddings) [28] or the triplet loss (three input - user prefers item 1 over item 2) [29]. We plan to address some of these issues in a future work.

REFERENCES

[1] Chang, S., Han, W., Tang, J., Qi, G.J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 119–128. ACM (2015)
[2] Dong, Y., Chawla, N.V., Swami, A.: Metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 135–144. KDD '17, ACM, New York, NY, USA (2017)
[3] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. CoRR abs/1607.00653 (2016), http://arxiv.org/abs/1607.00653
[4] Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. CoRR abs/1709.05584 (2017), http://arxiv.org/abs/1709.05584
[5] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 701–710. KDD '14, ACM, New York, NY, USA (2014)
[6] P. Cui, X. Wang, J. Pei, and W. Zhu: 2018. A survey on network embedding. In IEEE Transactions on Knowledge and Data Engineering, 2018
[7] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013), http://arxiv.org/abs/1301.3781
[8] https://www.pewresearch.org/fact-tank/2017/06/12/growth-in-mobile-news-use-driven-by-older-adults/
[9] Das, A. S., Datar, M., Garg, A., Rajaram, S. : Google news personalization: scalable online collaborative filtering, Proceedings of the 16th international conference on World Wide Web, 2007
[10] Yehuda Koren, Robert Bell and Chris Volinsky : Matrix factorization for recommender systems. https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf
[11] Konstan, J. A.,Miller, B.N.,Maltz,D.,Herlocker, J. L.,Gordon, L. R., And Riedl, J. Group-Lens: Applying collaborative filtering to usenet news. Commun. ACM 40, 77-87. 1997.
[12] Quoc V. Le, Tomas Mikolov : Distributed Representations of Sentences and Documents. Proceedings of the 31 st International Conference on Machine Learning (ICML), Beijing, China, 2014.
[13] Billsus, D., Pazzani, M. J.: User Modeling for Adaptive News Access, User Modeling and User-Adapted Interaction, v.10 n.2-3, p.147-180, 2000
[14] Tan, A. and Tee, C. : Learning User Profiles for Personalized Information Dissemination. Proceedings of 1998 IEEE International Joint conference on Neural Networks, pp. 183- 188, May 1998
[15] Carreira, R., Crato, J. M., Gon?alves, D., Jorge, J. A. : Evaluating adaptive user profiles for news classification, Proceedings of the 9th international conference on Intelligent user interfaces, 2004.
[16] Billsus, D., & Pazzani :M. A hybrid user model for news story classification. In Proceedings of the Seventh International Conference on User Modeling. 1999
[17] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M. : Combining Content-Based and Collaborative Filters in an Online Newspaper. In Proceedings of ACM SIGIR Workshop on Recommender Systems, 1999
[18] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, Jure Leskovec : Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time.
[19] William L. Hamilton, Rex Ying, Jure Leskovec: Inductive Representation Learning on Large Graphs, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
[20] Chong Wang and Chong Wang: Collaborative Topic Modeling for Recommending Scientific Articles, KDD11, August 2124, 2011, San Diego, California, USA.
[21] Christian Wartena, Wout Slakhorst, Martin Wibbels: Selecting keywords for content based recommendation, Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM), Toronto, ON, Canada October 26 - 30, 2010.
[22] Spacy - Industrial-Strength Natural Language Processing https://spacy.io/
[23] Janu Verma, Srishti Gupta, Debdoot Mukherjee, Tanmoy Chakraborty : Heterogeneous Edge Embeddings for Friend Recommendation, ECIR Cologne, April 2019.
[24] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology 58(7), 1019–1031 (2007)

[25] McInnes, Leland and Healy, John and Astels, Steve : hdbscan: Hierarchical density based clustering, The Journal of Open Source Software, Vol 2, number 11, 2017.
[26] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008
[27] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, Alex Alemi: Watch Your Step: Learning Node Embeddings via Graph Attention, 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montral, Canada.
[28] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf: DeepFace: Closing the Gap to Human-Level Performance in Face Verification, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2014.
[29] Florian Schroff, Dmitry Kalenichenko, James Philbin: FaceNet: A Unified Embedding for Face Recognition and Clustering, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2015/