EasyChair Preprint

№ 1254

Personalized high quality news recommendations using word embeddings and text classification models

Christos Samarinas and Stefanos Zafeiriou

June 30, 2019

Personalized high quality news recommendations using word embeddings and text classification models

Christos Samarinas
Donaco LTD
London, UK
chris.samarinas@gmail.com

Stefanos Zafeiriou
Imperial College London
London, UK
s.zafeiriou@imperial.ac.uk

Abstract

Reading news articles is one of the most important activities online and many apps have appeared the last few years for this purpose. In this paper, we present the architecture of a news recommendation system that provides personalized results to the users. We introduce a method to model the users' interests over time using word embeddings and a framework to filter and score high quality news stories using text classification models and agglomerative news clustering.

CCS Concepts • **Information systems → Recommender systems**; **Clustering and classification**.

Keywords word embedding, recurrent neural networks, text classification

1 Introduction

News articles are a form of online content that captures a large amount of Internet users' interest. The last few years they are widely consumed by mobile users via online social platforms. Consequently, there is an increased interest in promptly identifying the high quality articles that will receive a significant amount of attention and match the interests of each user. This complex task falls under the scope of news popularity prediction, click-bait detection and recommender systems.

A recommender system is an information filtering system that tries to predict the preference (for example rating) a user would give to an item [22]. The last few years recommender systems have become very popular. They are used in online marketplaces like Amazon, Google Play, Steam, streaming platforms like Netflix and obviously on social networks like Facebook and Twitter. There are three types of recommender systems: systems using *collaborative filtering*, *content-based filtering* and *hybrid systems* that combine the first two approaches [14]. *Collaborative filtering* is a technique that is used for making automatic predictions (filtering) about the interests of a user by collecting preferences from many users (collaborating) [26]. Users that have common items they like, get a recommended item from the other users. The data from the users' likes is used for making the recommendations. They make up a matrix of users and items, where each element is the rating of a specific item from a user. The users-items matrix is large and sparse. For this reason dimensionality reduction techniques are used like singular value decomposition and probabilistic latent semantic analysis. By reducing the dimensions of the users-items matrix, we also get item vectors with less dimensions and we can speedup a lot the recommendation process. *Content-based filtering* uses the description of the items and a profile of the user's preferences to make recommendations. Usually information retrieval techniques are used to match the profile of each user with similar items. More recent approaches use neural networks to perform the dimensionality reduction task in collaborative filtering as well as the content-based filtering. In this paper we present the design of a content-based news recommender system.

2 Related work

2.1 News recommender systems

A lot of research has been done the last few years in recommender systems and personalized news recommendations. Some systems use classic methods like collaborative filtering [10]. Google News has used collaborative filtering combined with a Bayesian framework that models the users' click behavior [18]. Another proposed system uses content-based filtering using keywords from the articles and constructs a profile for each user with these keywords [16]. In [20] the same method is used but the keywords are filtered using a neural network. The framework proposed in [29] uses ensemble hierarchical clustering that separates the users into different groups based on their reading histories, where each user might belong to several groups. For user profiling they use Latent Dirichlet Allocation, a model that is used very frequently for topic modeling [24]. In [9] they used context trees for news recommendations to anonymous visitors based on current browsing behavior. Sometimes, instead of a term-based approach, an ontology-based approach is adopted [13].

In some more recent work, recurrent neural networks are used for session-based news recommendations [11]. The sequence of viewed articles is given as input (the user's session) and article ids are given to the output (the recommendations). A variation of this approach adds an additional GRU recurrent layer to model information across user sessions and to track the evolution of the user interests over time [21]. YouTube uses a scalable end-to-end neural architecture for video recommendations using embeddings of videos and search tokens as input, and video probabilities as output [7].

2.2 Popularity prediction

Predicting the popularity of news articles is a challenging task. Many methods have been proposed and they depend on different types of data. The most common methods are regression models that depend on early user's access (like the number of comments the first few minutes) to predict the popularity [25] [23] [3]. A different approach uses a model of social dynamics to predict the popularity [17] based on early votes. In this paper, we use a different approach with a text classification model, which identifies headlines that are likely to attract many views.

2.3 Click-bait detection

Click-bait has become a very common phenomenon as it is an easy way for news publishers to make more profit. Click-bait articles are low quality articles that are designed to capture the users' attention. Click-bait headlines typically aim to exploit the "curiosity gap", providing just enough information to make readers curious, but not enough to satisfy their curiosity without clicking through to the linked content. Many methods have been proposed for the recognition of click-bait headlines. In [5] an SVM with N-gram features and other features like sentence structure and click-bait language achieved 93% accuracy. In [1] a convolutional neural network is used, which gets as input a sequence of word vectors and achieved 90% accuracy. The highest accuracy of 98% has been achieved with a bidirectional LSTM network with word embeddings and character embeddings as features [2].

3 System design

Figure 1 summarizes the architecture of the proposed content-based news recommender system. In general, the system consists of three processes. The first process, downloads articles from web pages, extracts the clean text, extracts terms from the clean text, clusters the article with other related articles and saves them to the database. The second process involves the scheduled training of two types of models; a content-based recommendation model, and a language model for generating word embeddings for the entities that appear in the news articles. In our current system, we use an unsupervised content-based filtering approach and the fastText model for learning keyword embeddings [4]. The third process involves the user. After we calculate a vector that captures his interests, we return a news ranking based on a personalized score from the recommendation model, a predicted popularity score, a coverage score and a click-bait score. After every click he makes on a news article, his profile of interests is updated, so that in the next session, he gets a new ranking based on his top most recent interests.

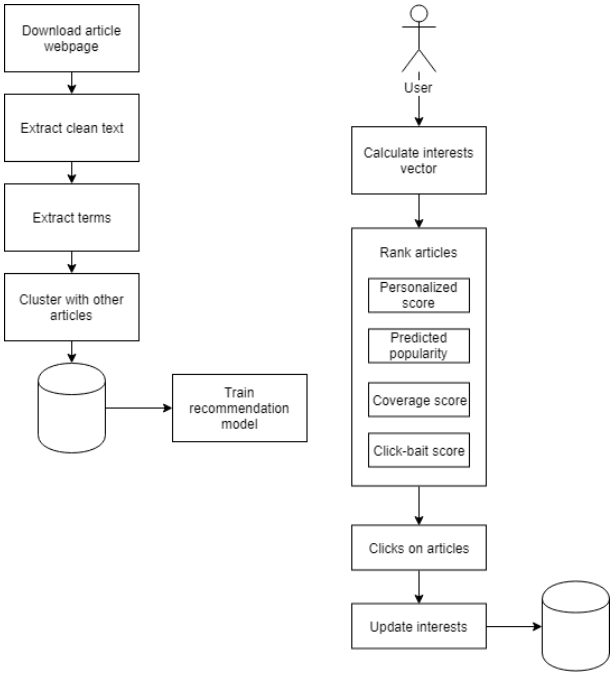

Figure 1. The processes of the news recommender system

4 Personalized results

4.1 Preprocessing

The first obvious step in a news recommender system, is the preprocessing of the downloaded articles. Firstly, we need to extract only the text from the HTML structure of the web page, ignoring the rest of the boilerplate content (navigation links, comments, ads). We extract the clean text using the algorithm proposed in [15], a decision tree to classify every HTML node based on link density and the number of words in each node. The clean text is then passed to the next step, the keywords extraction.

4.2 Keywords extraction

After extracting the clean text, we extract and score keywords from the content. More specifically, we extract nouns that may be preceded by adjectives. In order to extract these noun phrases, we do part-of-speech tagging using a fast averaged perceptron tagger [6] from the NLTK library [19]. Every article is represented by a set of keywords K, where each keyword k_i has a weight w_i based on its frequency and its similarity with the other extracted keywords. We calculate the weights using the graph-based approach proposed in [27]. For every pair of keywords k_i and k_j we calculate their attraction score as follows:

$$attr(k_i, k_j) = freq(k_i) \times freq(k_j) \times sim(k_i, k_j)$$

where $freq(k_i)$ is the frequency of k_i and $sim(k_i, k_j)$ the cosine similarity of the word embeddings of the keywords k_i and k_j. Using the attraction scores, we create a weighted

keyword	frequency	keyword	score
games	9	games	0.54
google	8	google stadia	0.46
pixel	8	stadia	0.29
stadia	8	pixel	0.26
google stadia	7	google	0.20
launch	3	phones	0.19
information	2	mobile gaming	0.05
things	2	stadia platform	0.03
platform	2	stadia controller	0.02
content	2	pixel phones	0.01

Table 1. Frequency & graph-based keywords ranking

undirected graph G where the vertices represent the keywords and the edges hold their attraction scores. The final weight of a keyword k_i is calculated as its weighted PageRank score in G:

$$w(k_i) = (1-d)+d\times \sum_{k_j \in C(k_i)} \frac{attr(k_i,k_j)}{\sum_{k_m \in C(k_j)} attr(k_j,k_m)} w(k_j) \quad (1)$$

In table 1 we can see an example of frequency-based ranking and how the graph-based scoring improves the results by using the semantic relationships between the entities. Terms that match with an entity on DBpedia are added to a database of entities (table 2) which is used to create interest profiles for the users.

id	name	description	...
1	Donald Trump	...	
2	Facebook	...	
3	Google	...	
	...	...	

Table 2. Database of entities

4.3 Entity embeddings

Periodically, we use a large representative sample of news articles to learn word embeddings for the different entities using the fastText model [4]. We do not use pretrained word embeddings, because the relationships between entities in the news articles change over time and also new entities can appear. For example, the entity 'Virtual Reality' was more related to the entity 'Facebook' when Facebook acquired the virtual reality company Oculus VR.

4.4 Content-based recommendations

In a news application we have users and articles, and we want to recommend interesting articles to each user. For this purpose we need a semantic vector representation for the articles based on their content and a semantic vector representation of the users based on the articles they read. Every article a is represented as the weighted average embedding of its keywords:

$$\overrightarrow{a} = \sum_{k \in V(a)} w(k) \cdot embedding(k) \quad (2)$$

where $V(a)$ is the set of extracted keywords (entities) from the article a and $w(k)$ the weight of the keyword k computed by (1). For each user u, we keep a profile of interests $P(u)$. Every interest has an id referring to an entity, the total number of clicks by the user on articles that contain it, the number of clicks on articles that contain it the last k days and a Unix timestamp of the last click.

uid	entity id	total clicks	recent clicks	last click
238	2	23	534	1536248321
238	3	4	23	1536241332
238	1	12	234	1536246374
	...	...	...	...

Table 3. Database of user interest profiles

Based on the number of clicks the last d days and the Unix timestamp of the last click, each user interest $k \in p(u)$ gets a decaying score over time:

$$s(k,u) = w_1 \cdot \frac{rc(k)}{max_{m \in P(u)} rc(m)} \cdot 2^{-\frac{t_p - k_t}{h}} + w_2 \cdot \frac{c(k)}{max_{m \in P(u)} c(m)} \quad (3)$$

where $rc(k)$ is the number of clicks of the interest k the last d days (recent clicks), $c(k)$ the total number of clicks of the interest k, k_t the Unix timestamp of the first click within the last d days, t_p the current Unix timestamp, and h a halving interval of the interest score measured in seconds. The first term in the score captures the importance of the entity for the user based on his recent activity the last d days, and the second term the overall importance of the entity based on his whole click activity. w_1 and w_2 are parameters which sum to 1 and define the importance of the recent and overall click activity of the user in the scoring of his interests. The user is represented by the weighted average of the word embeddings of his interests:

$$\overrightarrow{u} = \sum_{k \in P(u)} s(k,u) \cdot embedding(k) \quad (4)$$

After representing both the articles and the users in the same vector space, we need to calculate the similarity of each user with the articles. This similarity is the *personalized score* $ps(u,a)$ of the article a for the user u, and is calculated as the

cosine similarity of $\overrightarrow{a}$ and $\overrightarrow{u}$:

$$ps(u, a) = cos_\theta(\overrightarrow{u}, \overrightarrow{a}) = \frac{\overrightarrow{u} \cdot \overrightarrow{a}}{|\overrightarrow{u}||\overrightarrow{a}|} \quad (5)$$

5 Agglomerative clustering

One very important part of the news ranking is the clustering of the related articles that cover the same event.

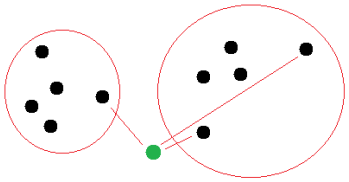

Figure 2. Hierarchical clustering of news articles

For this purpose, we use the following agglomerative clustering algorithm:

Algorithm 1 Agglomerative news clustering algorithm

1: Given a recently collected news article q, calculate its vector representation using (2)
2: Find the article a with the highest cosine similarity s_{close} with q with a publication time difference $dt < T$
3: Calculate the minimum cosine similarity s_{far} of q with the articles of the cluster of the article a
4: If $s_{close} > C$ and $s_{far} > F$ then add q to the cluster of a, otherwise create a new cluster with the article q

T, C and F are three thresholds that must be defined. We use these three thresholds, instead of just one, to avoid the phenomenon of *concept drift*, when the minimum similarity between articles in the same cluster becomes very low. The threshold T helps to prevent the concept drift, because articles with large publication time difference usually do not refer to the same event, or they refer to an update or progress of the same event. To speedup the second step of finding the article with the highest cosine similarity, we use an approximate nearest neighbors index [1].

6 News classification

In order to improve the quality of the personalized news results, we trained two text classification models for click-bait detection and popularity prediction. The click-bait detection model helps us filter out low quality articles that are not related to news events, and the popularity prediction model gives an estimate of the level of engagement of the article based on the headline. The two models are based on a Bi-LSTM neural network architecture.

[1]https://github.com/spotify/annoy

6.1 Bi-LSTM

The Long Short Term Memory networks (LSTMs) [12] are a special type of Recurrent Neural Networks (RNNs) designed to overcome the vanishing and exploding gradients problem of RNNs. LSTMs have additional memory cells which store memory from long distance terms. Because of their ability to capture long term dependencies in the data more effectively, they have been widely used in problems like language modeling, translation and speech recognition. An LSTM unit includes an input layer, a hidden layer and an output layer. At time t, it consists of inputs (i_t), output (o_t), forget gates (f_t) and memory cell (c_t). The updates at time t are then:

$$i_t = \sigma(W^{(i)}h_{t-1} + U^{(i)}x_t + b^{(i)})$$
$$f_t = \sigma(W^{(f)}h_{t-1} + U^{(f)}x_t + b^{(f)})$$
$$o_t = \sigma(W^{(o)}h_{t-1} + U^{(o)}x_t + b^{(o)})$$
$$\widetilde{c}_t = \tanh(W^{(c)}h_{t-1} + U^{(c)}x_t + b^{(c)})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t$$
$$\widetilde{h}_t = o_t \odot \tanh(c_t)$$
$$h_t = [\widetilde{h}_{t\rightarrow}; \widetilde{h}_{t\leftarrow}]$$

The last equation defines the output of the LSTM as the concatenation of a left to right and a right to left representation (bidirectional LSTM).

6.2 Click-bait detection

For training our click-bait detection model, we used two publicly available datasets. The first one is a collection of titles from the pseudo-news website The Examiner [2], containing click-bait headlines written by 21000+ authors over 6 years. This dataset does not necessarily contain only click-bait headlines, but in general headlines of very low quality news articles. The second dataset is the large collection of trustworthy high quality articles from Reuters over the last 10 years [3]. Combining the two datasets, we have a dataset of 2,500,000 headlines, where half of them are click-bait (low quality) and the other half non click-bait (high quality). With a single layer of 32 Bi-LSTM units, initialization of the word embedding layer with pretrained embeddings [4], softmax activation function in the output layer, categorical cross entropy loss and Adam as the optimization algorithm, we achieved 94.79% accuracy in our dataset (using 80% for training and 20% for validation). The probability score of the model is used for the filtering of the low quality click-bait articles.

6.3 Popularity prediction

For this task, we used a publicly available dataset from reddit, with 522,278 news titles with their number of upvotes,

[2]https://www.kaggle.com/therohk/examine-the-examiner
[3]https://github.com/philipperemy/Reuters-full-data-set
[4]https://fasttext.cc

downloaded from the worldnews subreddit [5]. We addressed the problem as a classification task. We tried to define a number of popularity classes but we got the best results with only two popularity classes; popular and not popular. In the popular class we put articles with at least 3000 upvotes and in the not popular class, articles with 0 upvotes. Using the same setup with the click-bait detection model, we achieved 78.5% accuracy. The score of this model is used to improve the ranking of the news articles that have been published very recently and are likely to be engaging.

7 Personalized ranking

Source quality

We define the quality of a news source s as follows:

$$sq(s) = w_1 \times originality(s) + w_2 \times hotness(s)$$

where $w_1 + w_2 = 1$, *originality* is the percentage of articles published by s that at the time of publication they had no related articles and *hotness* the percentage of articles published by s that had at least H related articles. The threshold H is a parameter that needs to be defined and depends on the total number of sources we have.

Coverage score

The coverage score of a news article a is the time-decaying popularity score based on the number of articles that cover the same news story (related articles in the same cluster) and the quality of their source:

$$cv(a) = 2^{\frac{t_p - t}{h}} \times \sum_{r \in R(a)} e^{sq(r)}$$

where $R(a)$ the set of the related articles to a (including a), t_p the current Unix timestamp, t the publication Unix timestamp of a and h a score halving interval in seconds. We use the exponential function so that the large differences between the source quality scores are more obvious, and articles that are covered by a few high quality sources get higher coverage score than articles that are covered by many low quality sources.

Filtered personalized ranking

The final personalized news score is defined as follows:

$$s(u,a) = \begin{cases} 0, & cb(a) > 0.5 \\ ps(u,a) \times cv(a) \times pp(a), & cb(a) \leq 0.5 \end{cases}$$

where $ps(u,a)$ is the personalized score of the article a for the user u, $cv(a)$ the coverage score of a, $pp(a)$ is the predicted popularity score of a and $cb(a)$ the click-bait score of a. If $|R(a)| > k$, then we set $pp(a) = 1$, so essentially we use the predicted popularity score only for articles with at most k related articles. News articles that have more than k related articles are considered popular, so we set their predicted popularity score to 1.

[5] https://www.kaggle.com/rootuser/worldnews-on-reddit

8 Conclusions

We presented the design of a content-based news recommendation system. We based our novel ranking framework on four scores; a personalized score using a semantic vector representation for the articles and the users, a click-bait score, a predicted popularity score based on the headline, and a coverage score based on the number of the related articles and the quality of their source.

9 Future work

One of the first improvements that can be made in the system is related to the text classification models. A fine-tuned pretrained language model like BERT [8] or XLNet [28] may give higher accuracy in both tasks. The popularity prediction model can be trained on a larger dataset, which can be constructed by crawling a large number of news sources and clustering the news articles. The popularity of every news article in the new dataset will be defined by the number of its related news articles. More research needs to be done on the *extraction of keywords* (entities) and their *disambiguation*. In some cases, we have an entity that can be different things based on the context. For example 'Isis' is an entity that can refer to the 'Islamic State of Iraq and the Levant', but it can also refer to the goddess in the ancient Egyptian religion or the 'River Thames'. Additionally, the system could also give results using a collaborative filtering approach. We could do dimensionality reduction on the users - article views matrix to get a low dimensional representation of the news articles and use it in our ranking framework.

References

[1] Amol Agrawal. 2016. Clickbait detection using deep learning. In *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on*. IEEE, 268–272.

[2] Ankesh Anand, Tanmoy Chakraborty, and Noseong Park. 2016. We used Neural Networks to Detect Clickbaits: You won't believe what happened Next! *CoRR* abs/1612.01340 (2016). arXiv:1612.01340 http://arxiv.org/abs/1612.01340

[3] Roja Bandari, Sitaram Asur, and Bernardo A. Huberman. 2012. The Pulse of News in Social Media: Forecasting Popularity. *CoRR* abs/1202.0332 (2012). arXiv:1202.0332 http://arxiv.org/abs/1202.0332

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *CoRR* abs/1607.04606 (2016). arXiv:1607.04606 http://arxiv.org/abs/1607.04606

[5] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE, 9–16.

[6] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 1–8.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[9] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized news recommendation with context trees. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 105–112.

[10] Florent Garcin, Kai Zhou, Boi Faltings, and Vincent Schickel. 2012. Personalized news recommendation based on collaborative filtering. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, Vol. 1. IEEE, 437–441.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[12] Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:https://doi.org/10.1162/neco.1997.9.8.1735

[13] Wouter IJntema, Frank Goossen, Flavius Frasincar, and Frederik Hogenboom. 2010. Ontology-based news recommendation. In *Proceedings of the 2010 EDBT/ICDT Workshops*. ACM, 16.

[14] Hosein Jafarkarimi, Alex Tze Hiang Sim, and Robab Saadatdoost. 2012. A naive recommendation model for large databases. *International Journal of Information and Education Technology* 2, 3 (2012), 216.

[15] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 441–450.

[16] Michal Kompan and Mária Bieliková. 2010. Content-based news recommendation. In *International conference on electronic commerce and web technologies*. Springer, 61–72.

[17] Kristina Lerman and Tad Hogg. 2010. Using a Model of Social Dynamics to Predict Popularity of News. *CoRR* abs/1004.5354 (2010). arXiv:1004.5354 http://arxiv.org/abs/1004.5354

[18] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized News Recommendation Based on Click Behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 31–40. https://doi.org/10.1145/1719970.1719976

[19] Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[20] Kyo-Joong Oh, Won-Jo Lee, Chae-Gyun Lim, and Ho-Jin Choi. 2014. Personalized news recommendation using classified keywords to capture user preference. In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 1283–1287.

[21] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[22] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[23] Gabor Szabo and Bernardo A Huberman. 2010. Predicting the popularity of online content. *Commun. ACM* 53, 8 (2010), 80–88.

[24] Serafettin Tasci and Tunga Gungor. 2009. LDA-based keyword selection in text categorization. In *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*. IEEE, 230–235.

[25] Alexandru Tatar, Panayotis Antoniadis, Marcelo Dias De Amorim, and Serge Fdida. 2014. From Popularity Prediction to Ranking Online News. *Social Network Analysis and Mining* (Jan. 2014), 4:174. https://doi.org/10.1007/s13278-014-0174-8

[26] Loren Terveen and Will Hill. 2001. Beyond Recommender Systems: Helping People Help Each Other. In *HCI in the New Millennium*. Addison-Wesley, 487–509.

[27] Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*, Vol. 39.

[28] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *ArXiv* abs/1906.08237 (2019).

[29] Li Zheng, Lei Li, Wenxing Hong, and Tao Li. 2013. PENETRATE: Personalized news recommendation using ensemble hierarchical clustering. *Expert Systems with Applications* 40, 6 (2013), 2127–2136.