Relation of Predictive Systems to User Reviews

Daniel Zhang and Victoria Caruso
University of Massachusetts Amherst
{danielzhang,vcaruso}@cs.umass.edu

ABSTRACT

The current research being examined on the topic of retrieval through user review is on how recommender systems are used to predict other items a user would like. These include models such as DeepCoNN and TransNets. Because these are the initial models, we aim to add a component of examining how applying word embedding techniques can improve these current review based prediction systems to retrieve the best information that can be used for future recommendations for that user.

1 RESEARCH QUESTIONS

1) How can user reviews of items and services be used to impact retrieval systems in a beneficial way?
2) In what ways does applying various word embedding techniques affect the results of model DeepCoNN in retrieving recommendations based on user reviews?

2 PROBLEM DESCRIPTION

One area that has been lacking in information retrieval techniques is the use of review texts as a heuristic in determining prediction ratings for queries involving a target user and a target item or service. In our research, we aim to explore how we can use review texts to characterise items or services as well as the use of word embedding models on DeepConn or Transnet models. Then using this information, we hope to improve the predicting ratings for related queries. This is relevant to our class because not only will we be using techniques learned in class to pre-process our data such as tokenization and normalization, but we will be using word embedding techniques that we have learned in class. We will also be examining the relationship between user ratings and the retrieval of recommended items and how these user reviews can improve information retrieval for queries involving a target user and target item.

3 MOTIVATION

Reccommender systems are important aspect of everyday life for most people that are browsing the internet or shopping online. They aid people in finding additional relevant articles, videos or products to the ones they are currently examining. Recommender systems not only assist people looking for more information or products but they also help companies to process all of their data and group ideas and products together. By using the component of user reviews on different products we believe we can make recommender systems more effective for both customers and companies. There are some user queries which will involve a target user searching for a target item of service. This involves knowing what properties the user is looking for as well as properties of a particular item or service. For example; we could have queries such as, "what laptop should I get as a student?" As an observer, we can make some assumptions about the properties of the laptop like the laptop needs to be portable and has a decent price for the value. Thus implementing a review text based search engine would allow users to receive recommendations based on properties that the search engine learns from review text.

4 RELATED WORK

In the paper Hidden Factors and Hidden Topics [1] , researchers explore how various aspects of a product or service can be reflected in the difference of reviews by different users. They tackle not only the aspects of a product or service, but how different users can perceive each aspect in a different manner, such as positive or negative. Ultimately, the paper looks more to undercover and explore the topic of using review text and review scores, while our paper focuses more on how we can improve search engines using review text.

Joint Deep Modeling of Users and Items Using Reviews for Recommendation [2] is similar to TransNets: Learning to Transform for Recommendation by Rose Catherine and William Cohen. This paper presents a new model called Deep Cooperative Neural Networks (DeepCoNN) which is made up of two parallel neural networks. The first of these networks prioritize learning the behaviors of individual users by exploiting the reviews written by the user being examined. The other network focuses on learning the specifications and characteristics of a reviewed item by examining the review itself. They then layer these networks to together to form the model DeepCoNN. They represented written reviews, using word embedding techniques through deep models while past techniques competing with DeepConn used a bag of words method. The researchers have found positive results for this method which have demonstrated DeepCoNN has outperformed all the baseline systems for recommendation that were currently being used at the time.

TransNets [3] focuses on the potential improvements of recommender systems by incorporating review texts into the predictive ratings. They claim that DeepCoNN is not as useful because it is only beneficial for the system when a review from the user for the target item is available which is not always the case in a real world scenario. To improve upon DeepCoNN, they add an additional layer to the model for when the review of a target item is not written by a target user. This additional layer is created with the purpose to approximate the review of the user and item pair.

Efficient Estimation of Word Representations in Vector Space[4] introduces the methodologies of Word2Vec. This includes two different architectures for word embedding including a Continuous Bag-of-Words, architecture, and a Skip-gram one. These computations were of low complexity compared to other symantic language methods which allows them to be found not only highly accurate but also useful for large scale datasets that can include an extremely large vocabulary size. These methods both were concluded to have

outperformed the previous state of the art methods of SemEval-2012.

The paper GloVe: Global Vectors for Word Representation[6] introduces another method of expanded vocabulary through using vector space models to represent language. They develop the method of GloVe, which uses matrix factorization techniques through the statistics of word occurrences in a given training corpus to find the relationship between terms and then use those terms most similar to expand the query.

5 DATA

5.1 Review Data

We will be using the Amazon Review Data. This data features a range of reviews of categories of items from May 1996 to October 2018. This data set has 233.1 million reviews total and includes reviews featuring ratings, text, and a vote, along with product information featuring the category, description, price, brand and image. We will be selecting smaller categories from this large data set including: Beauty, Kindle Store Subset, Clothing Shoes and Jewelry, Sports and Outdoor, and Pet Supplies. We have selected our topics to include ones that have been used in research of other models such as TransNet[3] so we can make a comparison of the models and how they can be improved using our word representation methods on the search.

We will also be using a Yelp data set which includes 8,635,403 reviews from 160,585 different businesses and 2,189,457 users. The review statistics include a star rating out of five stars and a text review.

We will randomly split each of our data sets into a smaller training, validation and then testing sets using a ratio of 80 : 10 : 10. This random split will include and equal amount of users from every time frame. The split will be done by first randomnly splitting the data set into two subsets one for the training set, one to be split further. The second set is then split in half making the validation and training sets.

5.2 Data Preprocessing

While our DeepCoNN model was already formatted to accept the Amazon Review Data, we had to apply preprocessing methods on the Yelp data in order for the model to accept it. This including splitting the data into a CSV format so that data from each review is put into one line containing: user id, item id, text review, and numerical rating. This required us to use normalization to eliminate any capitalization and punctuation in the review. Once the data was preprocessed we were able to use another preprocessor in the DeepCoNN model to handle this format.

5.3 Word Embedding Data

For training our word embedding models, we will be using the American National Corpus [1] which contains approximately 15 million words in English from contributions of spoken or written scrips from 1990 onward. We choose this as our corpus for training because the data comes from a vast amount of sources giving it diversity which include new forms of data such as emails and social media posts. This corpus is also free for use and does not hold any restrictions through the ANC website.

[1] www.anc.org/data/oanc/

6 METHODOLOGY

In this section we will be describing the approach used by DeepCoNN for text processing. The approach used for TransNet uses DeepCoNN as a basis and then expands upon it. Then we will be describing two word embedding models used for word embedding: Both methods of Word2Vec and Glove.

6.1 Word Embedding

We will be using two different methods for word embeddings. These are methods of word embedding are Word2Vec and GloVe. These methods work by finding semantically related terms to those that are in the search query and then expanding the query with those terms. We will be using the content of the target items review list to reform the original query q_{t-1} to our new query q_t. The methods of Word2Vec and Glove operate different from each other because Word2Vec uses log linear prediction methods to expand the query while GloVe uses matrix factorization techniques that rely on the count of words from the given corpus. While both models have been shown to be very effective using word embedding, the performance varies between the two depending on the different sized of the corpus which will result in varying vector sizes. [6]

6.2 Word2Vec

Word2Vec is a highly recognized method for word embedding which was originally created by Google using Google News as the training data set. Word embedding using Word2Vec are created using a feedforward neural network. The two common methods used to train the model are Skip-gram and Continuous Bag of Words (CBOW). It has been shown that CBOW is the faster model to train in terms of time and has slightly more accurate results for terms that occur more. Researchers have found that results of the Skip-gram model have shows that there is less training needed for the model to function well and that there is a higher performance for less frequent or rare terms. [5]

The CBOW model works by hiding a term and then using the neighboring terms to predict the hidden term. In the CBOW model, words are treated equally in terms of ordering which prevents the history of the positions from affecting how they will be projected [4]. This means that if we are to hide the middle word out of a sequence of five words, the first and the last word will be treated the same in terms of the influence of predicting the target term.

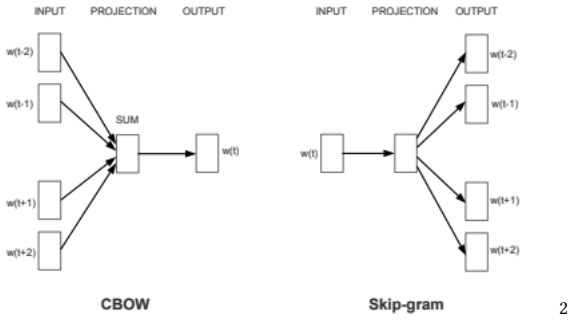
The Skip-gram model works in the opposite way by being given a single term and then using the model to predict the surrounding terms. The model uses a log classifier to calculate the range of words that will occur surrounding our input term. If we are given a query with $w_1, ..., w_n$ terms in which w_t is the target term, then the goal of the model to find the highest average of the probabilities given the corpus c that is used to train the model. [5]

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

The probability $p(w_{t+j}|w_t$ is calculated using the softmax function on the word w_t in its representation as a vector over all the words in the vocabulary given by the corpus. It had been found that using a larger corpus will create a higher accuracy but will also be costly

in terms of time to train the model.

We will be using Word2Vec to create word embeddings by finding words that are similar to each of the words in the search query and then expanding the original query to use those terms that score highest in similarity to the given term.



[2]

6.3 GloVe

GloVe model word embeddings are constructed through matrix factorization. It relies not only on local statistics of the corpus used to train but also uses global corpus statistics. GloVe operates by constructing a matrix of term co-occurrences based on their count. Within the matrix, the entry $X_{i,j}$ indicates the count that the term j will occur in the context of term i. [6] We then sum up all of the values of $X_{i,z}$ for all words z so we know the count that term i appears around all words, which is equal to the total count of word i. We then calculate the probability that a term j would appear in context of term i by dividing the count of occurrence of i with j, but the count of of i appearing with all terms.

$$P_{i,j} = Pj|i = X_{i,j}/\sum_{z=1} X_{i,z}$$

We can then find the similarity between words by comparing the ratio of these probabilities for different terms using different values of a probe term k. [6] The equation $P_{i,k}/P_{j,k}$ that produces a high number means that the probe term k will have high similarity with i, but not j. If the quotient is below low, the probe term has a high similarity with j, but not i. The model uses this as a basis by using these co-occurrence probabilities to find the relationships between terms. [6]

6.4 How we represent words

We will be using word embedding to represent the reviews as a matrix of word vectors using function $f : M \rightarrow \mathbb{R}^d$ where M is the size of the vocabulary and d is the number of dimensions of the vector. For a target user u, we concatenate all of the reviews written by that user into a document called d. This document will be denoted d_n^u where u is the user and n is the total number of words the user has written in reviews. We then construct a matrix of the word vectors for a user denoted as V_n^u by concatenating all of documents d_1^u through d_n^u for user u.

6.5 Convolution layer

After word embedding we then apply the convolution layer which will apply the convolution on each word vector V_n^u. It has an m amount of neurons that will apply a filter $d_j \in \mathbb{R}^{dxt}$ such that t is the amount of words in a window. For the i^{it} neuron, it will have features $z_i = f(V_n^u * K_i + b_i)$ where * is the convolution and b represents the bias. We then will take the feature with the highest value using pooling over all z_i: $o_j = max(z_i^1, ..., z_i^n)$. After we concatenate all of the neurons making the output: $O = [O_1, ..., O_n]$. This output is then used to create the output for both cases of users and items as $x = f(W \times O + g)$ where W is a weight matrix and g is the bias.[3]

6.6 Combining layers

In order to be able to map the outputs of users and items created in the convolution layer, we must construct a shared layer for the output of the user x_u and the output of the item y_j. We do this by concatenating these two outputs produced into a single vector z. We then use a second order model factorization machine (FM) which includes weight of the features to generate an estimate of the interaction between the user-item pair for each combination as follows:

$$J = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=1+1}^{|\hat{z}|} \langle \hat{v}_i, \hat{v}_j \rangle \hat{z}_i \hat{z}_j$$

$\hat{w}_0$ represents the global bias and w_i gives a weight to the each of the inputs. $\langle \hat{v}_i, \hat{v}_j \rangle$ represents the interactions between the users and the items as the inner product of their vectors. [2]

$$\langle v_i, v_j \rangle = \sum_{f=1}^{k} v_{i,f} \times v_{j,f}$$

6.7 Training

We will be training the FM using a minimization technique which takes the derivative of equation the with respect to J as done by the DeepCoNN model as follows.[2]

$$\frac{dJ}{d\hat{z}_i} = \hat{w}_i + \sum_{j=1+1}^{|\hat{z}|} \langle \hat{v}_i, \hat{v}_j \rangle \hat{z}_j$$

6.8 Experiment

We will be performing these experiments using each type of embedding. Each trial will include one of the following embeddings: no embedding, glove word embedding, cbow word2vec embedding, and sg word2vec embedding.

In order to obtain the glove word embedding we used the glove source code provided on github [3]. In order to run this code we needed a word frequency text file which we created from the OANC corpus [4]. Next to create the CBOW and SG word embedding we used the gensim's api [5] for creating word2vec embeddings. Here we used that same word frequency text file as well as an additional text file which lists every word in a document on each line which we created from the same corpus. Then to run the DeepCoNN model with word embeddings we used a modified version of the DeepCoNN code provided by winterant [6] . We modified the code to allow the use of no word embeddings as well as updating the config to run all four word embeddings on a data set.

[2] https://arxiv.org/pdf/1301.3781.pdf

[3] https://github.com/stanfordnlp/GloVe
[4] https://www.anc.org/da
[5] https://radimrehurek.com/gensim/models/word2vec.html
[6] https://github.com/winterant/DeepCoNN

7 EVALUATION METRICS

We will be evaluating our system by using Mean square Error (MSE) where N is the total datapoints, r_i is the true rating and $\hat{r}_i$ is our predicted rating. A higher value means a worse performance while a lower value indicated a better performance. We will compare our values of MSE to those given by past trials of DeepCoNN on reviews of Yelp data and TransNet on reviews of Yelp data.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(r_n - \hat{r_n})^2$$

8 RESULTS

In our experiment we present the use of word embeddings in conjunction with DeepCoNN. DeepCoNN model combines user reviews and product or service rating in order to create a better recommender system. A word embedding are representations for text, typically represented as a vector, with words that have similar meaning having similar vector representation. In our experiment we used four different word embeddings: a control with no word embeddings, the Glove word embedding, the continuous bag of words (CBOW) word embedding and the Skip-gram (SG) word embedding. Our results showed that using some type of word embedding typically improved mse results and those that performed worse only performed marginally worse than using no embedding at all. We found that the Glove word embedding outperformed every other embedding in every case and the CBOW word performing similarly in most cases.

Dataset	No Word Embedding	Glove
AZ - All Beauty	0.745584	0.670999
AZ - Kindle Store Subset	0.81839	0.710059
AZ - Clothing Shoes	1.21634	1.19815
AZ - Sports and Outdoors	0.875822	0.862099
AZ - Pet Supplies	1.396489	1.388607
Yelp	0.82139	0.730467
Dataset	**CBOW**	**Skip-Gram**
AZ - All Beauty	0.754326	0.617549
AZ - Kindle Store Subset	0.745701	0.707167
AZ - Clothing Shoes	1.216755	1.216224
AZ - Sports and Outdoors	0.871726	0.878858
AZ - Pet Supplies	1.39677	1.382166
Yelp	0.768362	0.768471

8.1 Word2Vec

We have found that the Word2Vec model on average performed better than using no word embeddings at all. However some Word2Vec models did perform worse, but only marginally.

8.1.1 CBOW. Our experiments showed that CBOW performed similarly to the other word embeddings, but did the worst out of the three embeddings. In most of the datasets CBOW had similar mse values to Skip-Gram and slightly worse values than Glove, however, for the AZ - All Beauty dataset, CBOW did much worse than Glove and Skip-Gram as well as being outpeformed by the use of no word embeddings.

8.1.2 Skip-Gram. As mentioned previously, Skip-Gram performed similarly to CBOW, but did outperform in a couple of datasets. Skip-gram was also better than no word embeddings in about half of the datasets. Skip-Gram tended to worse than the Glove word embedding, but never was outperformed by CBOW or No Word Embedding by any significant amount.

8.2 GloVe

For all of our testing we have found that Glove Word Embedding has lead to the best results for the DeepCoNN model. Each of our MSE values for all of our tested Amazon datasets and the Yelp dataset has shown a better performance with less error. We also experimented on using the GloVe pretrained model that was trained using Wikipedia data from 2014. We ran this trained model on two Amazon Datasets: AZ - Kindle Store Subset and AZ - Clothing. We found that the pretrained model performed similarly to the glove model created using the OANC corpus in the AZ - Kindle Store Subset, but slightly outperformed in the AZ - Clothing Shoes dataset. [7]

8.3 Comparison with TransNet

While our datasets used were more up to date, when comparing our statistics to those performed by Rose Catherine and William Cohen [3] which showed similar results of 1.2106 MSE for DeepCoN on Yelp17 data while for TransNet method they had 1.5913 MSE. While our statistics cant fully be compared to those because they are based on different datasets due to ours being more recent and thus being a super-set of their dataset, we believe that since we also ran DeepCoNN without word embedding, and received similar values, that these word embedding methods can be used to improve both models.

9 CONCLUSION

Because we have found the most positive results for GloVe this we think that using the GloVe model in future iterations of examining how user text reviews can be used to improve predictive recommendations. For the current DeepCoNN and TransNet models, they relied on representing the words as vectors using the Word2Vec. While this is good for the representation of the words as vectors, we think for expanding the query the matrix factorization methods of GloVe should be further examined and researched because of their optimistic results. To address our questions, we believe that overall user reviews of items and services can greatly impact retrieval systems in a beneficial way by allowing for more specific recommendations that are geared to each users taste based on the written recommendations from themself and other users. Lastly,we found that Word embeddings further improve the recommender system compared to the use of no word embeddings.

10 TEAM WORK

Each member ran queries on the different data sets. Daniel will be handing the query and analysis for the Amazon, while Victoria examines Yelp data. We will discuss the different outcomes of the two trials and each write sections of the report that the other will proofread.

[7] https://nlp.stanford.edu/projects/glove/

REFERENCES

[1] Julian McAuley and Jure Leskovec. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13). 165–172.
[2] Lei Zheng, Vahid Noroozi, and Philip S. Yu.. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17). 425–434.
[3] Catherine, R., and Cohen, W. 2017. TransNets: Learning to Transform for Recommendation. Proceedings of the Eleventh ACM Conference on Recommender Systems. 288–296.
[4] Mikolov, T., Corrado, G., Chen, K. and Dean J.. 2013. Efficient Estimation of Word Representations in Vector Space. In ICLR Workshop.
[5] Mikolov, T., Corrado, G., Sutskever, I., Chen, K. and Dean J.. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Advances in neural information processing systems. 3111–3119.
[6] ennington, J., Socher, R. (2014)]bib6 Pennington, J., Socher, R., Manning, C.. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 1532–1543.