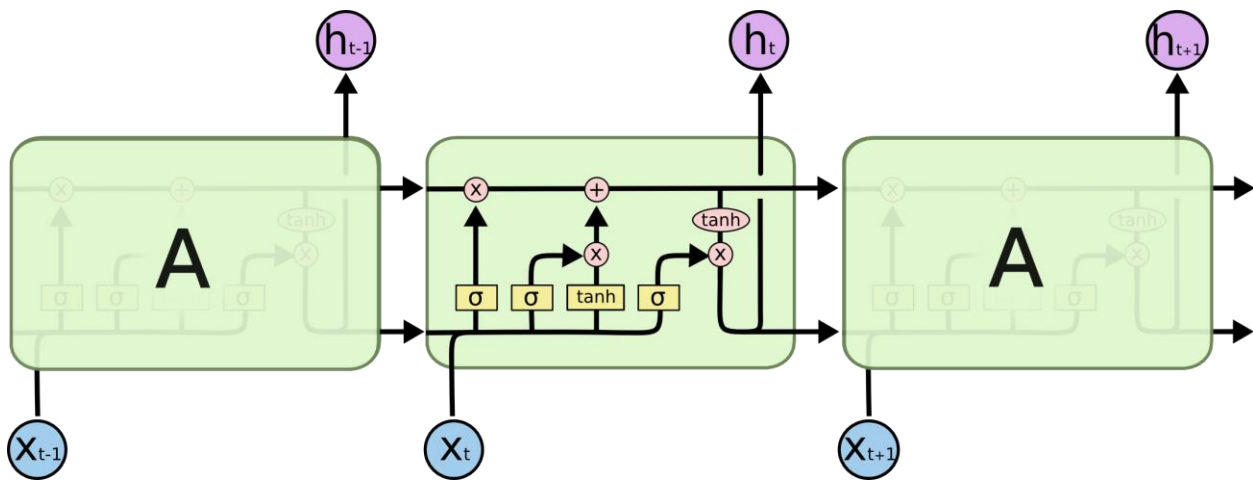


VIRTUAL RECOMMENDER

REPORT

Team 26



Instructor: Dr. James Wang

Pratyush Singh

Vrunal Mhatre

Clemson University

20.04.2017

CPSC 8650

Contents

1. Abstract
2. Proposal
3. Introduction
4. Motivation
5. Preliminary research
6. Environment Setup
7. Chatbot
 - a. Data Preprocessing
 - b. Word2Vec
 - c. Sequence to Sequence model
 - d. Long Short Term Memory (LSTM)
8. Recommendation System
 - a. Graph based community detection technique
 - b. Hierarchical clustering
 - c. Collaborative filtering
9. Evaluation and Results
10. Conclusion
11. Dataset links
12. References

1. ABSTRACT

Recommender System has seen meteoric rise since explosion of data to correlate user behaviour and trends. Through our project, we are able to use recommendation results to build a end-to-end application trained on Amazon product rating and review data sets. Users can chat with bots and gain insightful suggestions through the recommendation system. Our project converses with users by predicting the next sentence given the previous sentence or sentences in a conversation. As our chatbot is based on deep learning framework using sequence to sequence model instead of rule based approach, we find our chatbot is able to respond more intuitively and provide a direct medium for consuming suggestions.

2. PROPOSAL

Feb 2, 2017:

Recommendation system based on user reviews and feedback: A model trained on previous user reviews which could help recommending new queries/ customer in making insightful choices. Also promoting advertising for say Amazon. Our initial attempt would be to train a recommendation model (Collaborative system) on Amazon user review data. Extended project goals will involve building a chatbot which any user can talk, and find answers and suggestion on their queries

3. INTRODUCTION

Today there are many service providers who have been building there chatbot for providing assistance to their customers and to solve their queries. so we thought that if there could be possibility if the e-commerce companies can also build the chatbot to provide recommendations to their users. We have found ourselves browsing internet for shopping without having any goal in mind. Often we don't get correct recommendation for what we are searching for on these e-commerce web sites. So if there is a chatbot who has been trained to respond such queries will be of great use. Based on the dataset of amazon items that contain attributes about every item we have trained our Chatbot and recommendation system.

4. MOTIVATION

We followed up a challenge posted by Udacity Deep Learning course instructor, Siraj Raval [7] to build chatbots using movie dialogue corpus. We went above and beyond the challenge to build a recommender system integrated within a chatbot

5. Preliminary Research

Rule based model:

AIML (Artificial Intelligence Markup Language); an XML based “language” that lets developers write rules for the bot to follow. Like Siri in its early years tried to give predefined answers to questions. This wasn’t quite natural as it used to mine for response from rule-based mapping

One-hot Representation:

For Chatbot, we evaluated using one-hot encoding for our tokenized words in corpus. However, word2vec edges in sense of semantic and context mapping which cannot be done using one-hot encoding. One-hot encoding represents a word with an id in a sparse column matrix and difficult to use with machine learning models for text mining.

Choosing distance measures:

In collaborative filtering we have tested many distance measures to identify which is appropriate for us. Distance measure such as Euclidean distance and Manhattan does not works well in our case since our data is very sparse. We have also tested Pearson correlation coefficient because we were aware with the fact that users have different grading behavior and Pearson correlation coefficient measures correlation between users but the problem here was that it also considers every 0-0 pair. Since we wanted make an accurate prediction we have decided to go for cosine similarity but with a variation because cosine similarity also has a drawback of grade inflation. We finally agreed on adjusted cosine similarity which subtracts the user average from each correlated pair.

Similarity measure for community detection:

We had trouble in identifying communities since the size of communities was unknown to us. We had used jaccard similarity to find the clique in the graph but graph partitioning was difficult so we agreed to use networks maximum cut algorithm.

6. Environment Setup

1. Tensorflow 0.12, deep learning models for chatbot
2. Tensorboard for data visualization
3. Anaconda for python imports
4. Palmetto cluster with GPU processing.
5. Community Module

7. CHATBOT

Chatbots are conversational agents to communicate and absorb information for a particular application. We are using deep learning ensemble techniques to build natural language processing bots. Our target was to build a closed domain conversational agent which mines context related to a particular topic, Amazon product reviews. Many companies have started to integrate chatbot in third party messaging applications to interact with users directly via bots and in future we estimate chatbots will be incorporated as an important source of interaction with users.

Retrieval-based-Models:

We are using Deep Learning and Machine translation concepts to use ensemble classifier techniques. Sequence2Sequence and LSTM helps to improve context mapping within the conversations and yield natural response using word embedding.

Closed-domain:

Response to conversations will be specific to shopping assistant based on Amazon review dataset which is supported by our recommender system.

7.a DATA PREPROCESSING:

```
_PAD = b"_PAD"
_GO = b"_GO"
_EOS = b"_EOS"
_UNK = b"_UNK"
_START_VOCAB = [_PAD, _GO, _EOS, _UNK]

PAD_ID = 0
GO_ID = 1
EOS_ID = 2
UNK_ID = 3

_WORD_SPLIT = re.compile(b"([.,!?\\'\":;(){}])")
_DIGIT_RE = re.compile(br"\d")

def basic_tokenizer(sentence):
    words = []

    for space_seperated_fragment in sentence.strip().split():
        words.extend(re.split(_WORD_SPLIT, space_seperated_fragment))
    return [w for w in words if w]
```

```
'''
Create a dictionary with ( key = line_id, value = text )
'''

def get_id2review():
    reviews = open('amazon_user_review.txt').read().split('\n')
    id2review = {}
    for review in reviews:
        _review = review.split(' +++$+++ ')
        if len(_review) == 5:
            id2review[_review[0]] = _review[4]
    return id2review
```

Fig [1]. Parsing review data into test and training files in data_preparation.py

Fig.[2]: Bucketing and Padding data with keywords in myData_Utils.py

For Chatbot, we use Amazon review dataset and Cornell movie dialogue corpus for rich conversational data.

1. Convert raw sentences (user reviews) into tokenized ids
2. We split our data into encoder and decoder and tokenize each words to an id to help data retrieval faster.
3. We do the second step for training and testing data set and store as train.enc, test.enc and train.dec, test.dec

7.b Word2Vec

The purpose and usefulness of Word2vec is to group the vectors of similar words together in vector space. Neural word embedding represents a word with numbers of a certain fixed size, we used vectors of size 256. Hence, each word is a feed into a layer of neural net of size 256. We use the Google's pretrained Word2Vec library from Tensorflow to get vector values for each word

“One simple way to evaluate embedding is to directly use them to predict syntactic and semantic relationships like king is to queen as father is to __? “- man. This is called *analogical reasoning* and the task was introduced by Mikolov et al [3].

7.c Sequence to Sequence Model

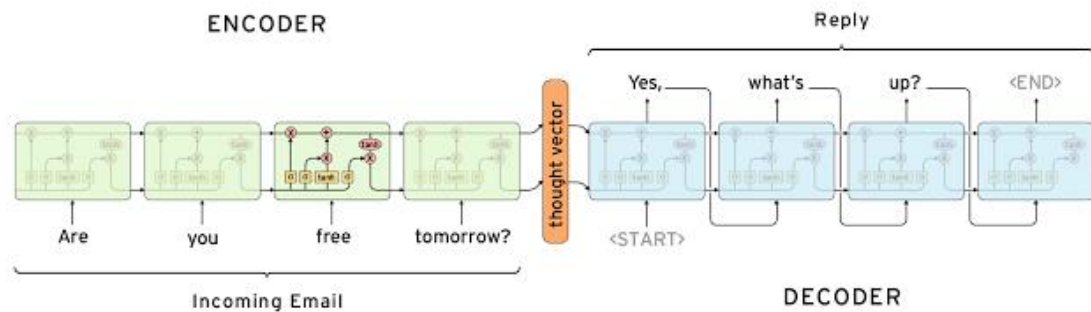


Fig.[3]: Sequence to Sequence model with thought vector and encoder and decoder networks

The main objective of sequence to sequence model are:

1. Interpret language
2. Track the context of conversation
3. Generate a response

Model contains two recurrent neural network encoder and decoder. An Encoder and a Decoder. The encoder takes a sentence as input and processes one word at each timestep. The `tf.nn.seq2seq` library from tensorflow internally converts word texts into vectors using `word2vec`.

To convert a sequence of symbols into a fixed size feature vector that encodes only the important information in the sequence while losing the unnecessary information. Each hidden state influences the next hidden state and the final hidden state can be seen as the summary of the sequence.[4] [Understanding LSTM - Chris Olah]

This state is called the context or thought vector as seen in fig.(3), it represents the intention of the sequence. From the context, the decoder generates another sequence, one word at a time. Here, at each time step, the decoder is influenced by the context and the previously generated symbols.

```
def seq2seq_f(encoder_inputs, decoder_inputs, do_decode):  
    return tf.nn.seq2seq.embedding_attention_seq2seq(  
        encoder_inputs, decoder_inputs, cell,  
        num_encoder_symbols = source_vocab_size,  
        num_decoder_symbols = target_vocab_size,  
        embedding_size = size,  
        output_projection = output_projection,  
        feed_previous = do_decode)
```

Fig.[4] : Tensorflow libraries in myseq2seq_model.py, with arguments for encoder and decoder inputs

7.d Long Short Term Memory(LSTM):

LSTM can handle and remember large gaps within the context of conversations for relevant information and the target. So LSTM are capable of handling long term dependency. Connects hidden layer to itself, using time-delayed connections. This allows the recurrent model to form some kind of short term memory, as information from the past can be represented by the hidden layer state that gets updated based on the current input and the state of the hidden layer in the previous time step. LSTM have a chain structure where instead of having single neural network they have 4 interacting in different way. LSTM cells remove or add information to their structure using gates (sigmoid functions with pointwise operation: Sigmoid value 0: remove everything 1: let everything out) [4]

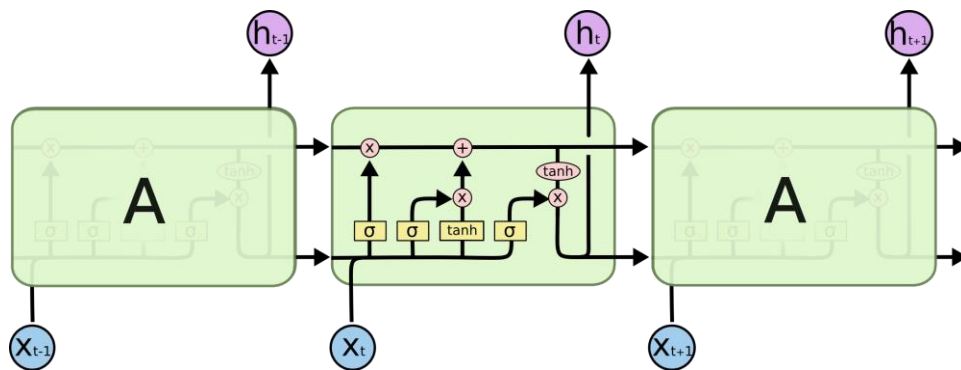


Fig.[5]: LSTM Pipeline

8 RECOMMENDER SYSTEM:

Recommendation system is a tool which is used by every other e-commerce companies to provide recommendations to their customers for their products on the basis of his past purchasing history. There can be many techniques to build a recommendation system. It is important to consider the nature and structure of dataset before approaching to the problem.

```
{
  "reviewerID": "A2XVJBSRI3SWDI",
  "asin": "0000031887",
  "reviewerName": "abigail",
  "helpful": [0, 0],
  "reviewText": "Perfect red tutu for the price. I bought it as part of my
                  daughters halloween costume and it looked great on her.",
  "overall": 5.0,
  "summary": "Nice tutu",
  "unixReviewTime": 1383523200, "reviewTime": "11 4, 2013"
}
```

Fig.[6]: Amazon data set [1]

```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-
amazon.com/images/I/51fAmVkJbyL._SY300_.jpg",
  "related": {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00DOWDS9A", "B00D0GCI8S",
"0000031895", "B003AVKOP2", "B003AVEU6G", "B003IEDM9Q",
"B002R0FA24", "B00D23MC6W", "B00D2K0PA0", "B00538F5OK",
"B00CEV86I6", "B002R0FABA", "B00D10CLVW", "B003AVNY6I",
"B002GZGI4E", "B001T9NUFS", "B002R0F7FE", "B00E1YRI4C",
"B008UBQZKU", "B00D103F8U", "B007R2RM8W"],
    "also_viewed": ["B002BZX8Z6", "B00JHONN1S", "B008F0SU0Y",
"B00D23MC6W", "B00AFDOPDA", "B00E1YRI4C", "B002GZGI4E",
"B003AVKOP2", "B00D9C1WBM", "B00CEV8366", "B00CEUX0D8",
"B0079ME3KU", "B00CEUWY8K", "B004FOEEHC", "0000031895",
"B00BC4GY9Y", "B003XRKA7A", "B00K18LKP2", "B00EM7KAG6",
"B00AMQ17JA", "B00D9C32NI", "B002C3Y6WG", "B00JLL4L5Y",
"B003AVNY6I", "B008UBQZKU", "B00DOWDS9A", "B00613WDTQ",
"B00538F5OK", "B005C4Y4F6", "B004LH21NY", "B00CPHX76U",
"B00CEUWU2C", "B00IJVASUE", "B00GOR07RE", "B00J2GTM0W",
"B00JHNSNSM", "B003IEDM9Q", "B00CYBU84G", "B008VV8NSQ",
"B00CYBULSO", "B00I2UHSZA", "B005F50FXC", "B007LCQI3S",
"B00DP68AVW", "B009RXWNSI", "B003AVEU6G", "B00HSOJB9M",
"B00EHAGZNA", "B0046W9T8C", "B00E79VW6Q", "B00D10CLVW",
"B00B0AVO54", "B00E95LC8Q", "B00GOR92SO", "B007ZN5Y56",
"B00AL2569W", "B00B608000", "B008F0SMUC", "B00BFXLZ8M"],
    "bought_together": ["B002BZX8Z6"]
  },
  "salesRank": {"Toys & Games": 211836},
  "brand": "Coxlures",
  "categories": [{"Sports & Outdoors", "Other Sports",
"Dance"}]
}
```

Fig.[7]: Metadata of amazon dataset: [2]

We are using amazon dataset which we have got through Julian McAuley from university of california san diego for educational purpose. This is a preprocessed dataset

which also has metadata which also contain information about relations (items which are bought_together , also_bought and also_viewed) between items. The dataset is very sparse since all users does not tend to rate every item in the dataset.Considering our dataset we decided to implement three techniques to build our recommendation system which uses different attributes of our dataset.

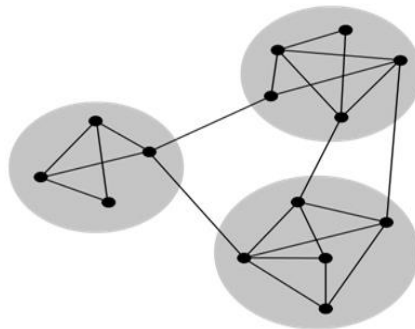
We have implemented three techniques:

1. Graph based community detection
2. Hierarchical clustering
3. Collaborative filtering

We have used metadata of our dataset to test graph based community detection technique and Hierarchical clustering technique and rating dataset to test our collaborative filtering technique. Using these techniques we are able to build a high quality recommendation system for millions of products.

8.1 Graph Based community detection:

Community are basically set of nodes that form a group such that each node in the group is densely connected to each other. The graph divide itself naturally into groups of nodes which are densely connected internally and sparsely connected between groups.



Community in graph:

Fig.[8]: https://en.wikipedia.org/wiki/Community_structure

To build a graph we have used Networkx libraries in python. Networkx comes handy with python packages like winpython and anaconda. We have build a undirected graph where each node in a graph represent item in our metadata and edges represent the relation between these items i.e if the item is in a related list of metadata (bought_together, also_bought, also_viewed) then there is an edge between those items.

The edges also has weights which represent the extent of relation between those nodes. We have given weights for each relation type we thought that relation bought_together is more important so we are considering its weight to be 1.0, heuristically we have also assigned the weight 0.3 to also_bought relation and 0.1 to also_viewed relation.

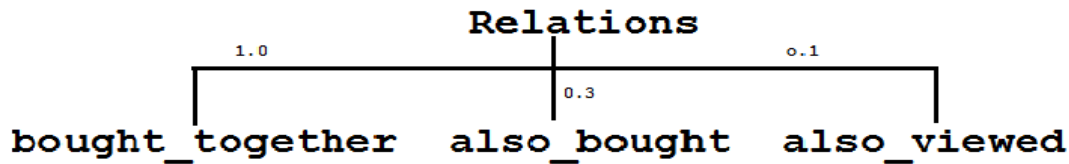


Fig.[9]: Relation weights

Finding communities was challenging, in order to find communities we need to define similarity measure for vertex clustering. Jaccard similarity was a good candidate for this, since our objective was to identify the clique in the graph. We have used this to identify nodes which are densely connected to each other. The problem we faced was graph partitioning, since we do not know number of community and the size of community. moreover size of community can be of different size. To solve this we have imported networkx community module and used its maximum cut (best_partition) algorithm to identify communities in the graph.

Implementation:

1. Build graph using Networkx functions.
 - a. For each asin in metadata add new node in graph.
 - b. For each asin in related list of metadata add an edge in graph.
 - c. Add weights respective to the relation type.
2. Find community in graph using maximum cut algorithm.
3. Give recommendation from the cluster of test item.

8.2 Hierarchical clustering:

Hierarchical clustering is a bottom up clustering method. This method also uses the same metadata. The clustering is done on the basis of relations (bought_together, also_bought, also_viewed) between items. Here also we have given weights for each relation type. For bought_together we are considering its weight to be 1.0, also_bought relation to be 0.3 and 0.1 to also_viewed relation. Here we use a distance measures as weight of relationship between clusters.

Implementation

1. Built a distance matrix which has sum of weight of relation between the items at each entry [item1, item2].
2. Update the matrix till it reaches the threshold:
 - a. Set zero for all distances of one of two clusters and updating the other.
 - b. Every entry in the matrix has to be set to zero which has [i,j] coordinates equal to the deleted clusters.
 - c. For the other clusters we use the mean between the existing distances.
 - d. If the item has relation with both of them then we put the average.
 - e. If the item has relation with only one of them then we put the half of that value.
3. Give recommendation from the cluster of test item.

8.3 Collaborative filtering:

We are using collaborative filtering based on ratings given by the user to amazon products. We use rating dataset for this method. Here we make recommendations decisions for a specific users based on judgments of users with similar interests. To identify the training users that share similar interest as the test user we need to find the k nearest neighbors (k-NN).

We need to define appropriate distance metric to compute the similarity between two users. Choosing distance metric was a crucial. Simple distances like Euclidean and Manhattan works well when the data is dense since are data is sparse because most users rate only few items we cannot use them. One thing to keep into mind is different users may have different rating behavior- this issue can be handled by Pearson correlation coefficient since Pearson correlation measures the correlation between the users but it also account 0-0 user rating pair. Our objective was to make accurate predictions so we have avoided 0-0 matches since it can give false positive. So we have used adjusted cosine similarity which subtract the user average from each correlated pair.

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_u)^2}}$$

Fig.[10]: Adjusted cosine similarity

Once we have identified nearest neighbor we can predict the ratings for the test user as the average of ratings by the training users with similar interest.

9. EVALUATION AND RESULTS

Language model evaluation method

Perplexity: The quality of language models is measured based on their ability to learn a probability distribution over words in datasets. Word embedding models are often evaluated using perplexity, a cross-entropy based measure. So, How good is our language model for chatbot

```
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:259] Raising pool_size_limit_ from 5305 to 5835
global step 48900 learning rate 0.4568 step-time 0.27 perplexity 5.02
eval: bucket 0 perplexity 11724.26
eval: bucket 1 perplexity 10924.70
eval: bucket 2 perplexity 5783.13
eval: bucket 3 perplexity 5515.28
█
```

Connected to user.palmetto.clemson.edu

Fig.[11]: Seq2Seq model training

Perplexity is the inverse probability of the test set, normalized by the number of words: Minimizing perplexity is the same as maximizing probability. A better model of a text is one which assigns a higher probability to the word that actually occurs. From our results we can see results for Uni-gram, bi-gram and tri-gram probabilities from which we can derive Maximum Likelihood Estimates (MLE) for a word in an unseen data set. With each epoch our perplexity has reduced, proving maximized probability.

```

Instructions for updating:
Please use tf.global_variables instead.
Reading model parameters from working_dir/seq2seq.ckpt-48600
> Hello World
Hello .
> How are you?
Okay .
> Who are you?
I am the _UNK .
> Time to answer tough questions
What ' s the matter , Baxter ?
> You are getting examined
_UNK ! !
> You don't know the answers to anything
No . . .
> Stupid
Have you ever met .
> Yes, i met you
_UNK , yes , sir !
> Funny!
What ? !
> You
Have you been enough ?
> Yes, But you are funnier than me
I ' m afraid I ' m not leaving you .
>

```

Connected to user.palmetto.clemson.edu

Fig.[12] Testing on our trained seq 2 seq modeled chatbot to have natural conversations and suggestions

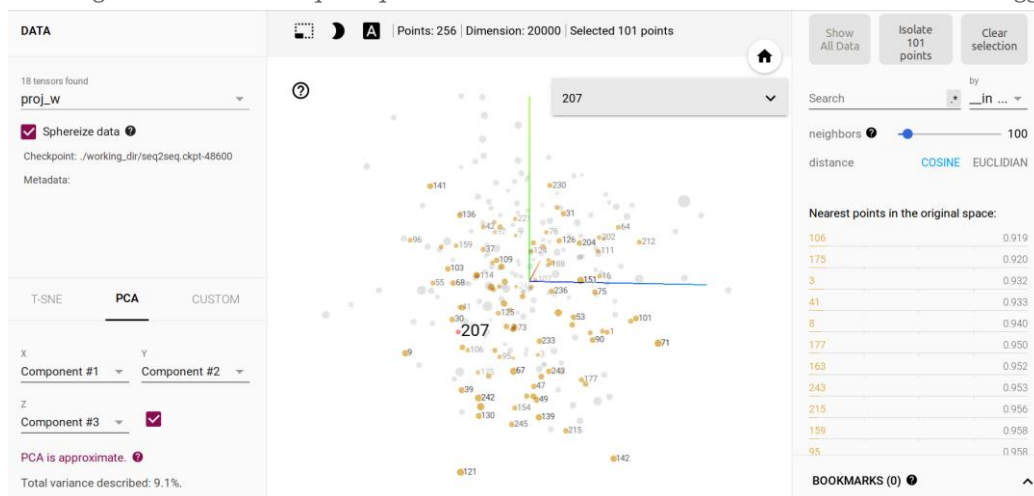
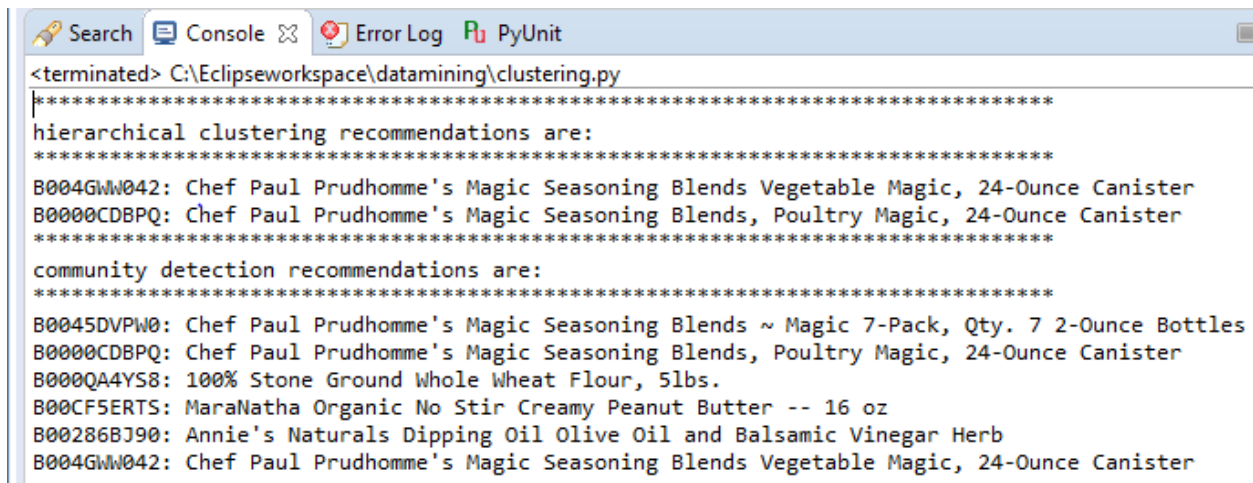


Fig.[13]: Word2vec Visualization on tensorboard for word similarity using cosine similarity.
Words that are similar end up clustering nearby each other

Result of clustering algorithms for recommendation system:

Test item: B0000CDBQT: Chef Paul Prudhomme's Magic Seasoning Blends Meat Magic, 24-Ounce Canister

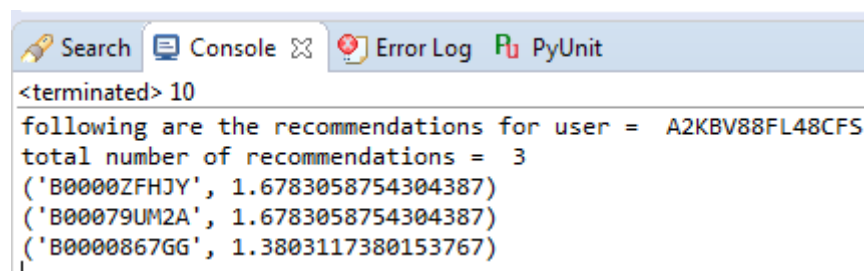


```
<terminated> C:\Eclipseworkspace\datamining\clustering.py
*****
hierarchical clustering recommendations are:
*****
B004GWW042: Chef Paul Prudhomme's Magic Seasoning Blends Vegetable Magic, 24-Ounce Canister
B0000CDBPQ: Chef Paul Prudhomme's Magic Seasoning Blends, Poultry Magic, 24-Ounce Canister
*****
community detection recommendations are:
*****
B0045DVPW0: Chef Paul Prudhomme's Magic Seasoning Blends ~ Magic 7-Pack, Qty. 7 2-Ounce Bottles
B0000CDBPQ: Chef Paul Prudhomme's Magic Seasoning Blends, Poultry Magic, 24-Ounce Canister
B000QA4YS8: 100% Stone Ground Whole Wheat Flour, 5lbs.
B00CF5ERTS: MaraNatha Organic No Stir Creamy Peanut Butter -- 16 oz
B00286BJ90: Annie's Naturals Dipping Oil Olive Oil and Balsamic Vinegar Herb
B004GWW042: Chef Paul Prudhomme's Magic Seasoning Blends Vegetable Magic, 24-Ounce Canister
```

Fig.[14]: Output screen of clustering algorithms.

The results of recommendation system is exciting when we give B0000CDBQT as input number which is for chef paul prudhomme's meat magic, we got recommendations for similar brand of vegetable magic and poultry magic for both hierarchical clustering and community detection. Moreover, in community detection we got more recommendations than hierarchical clustering. Further investigating on this we found the reason for this is because in a graph there are overlapping of community and few nodes are shared between communities. The results make sense because item B0000CDBQT has multiple relations with B004GWW042 and B0000CDBPQ and also connected to other recommendations.

Result of collaborative filtering for recommendation system:



```
<terminated> 10
following are the recommendations for user = A2KBV88FL48CFS
total number of recommendations = 3
('B0000ZFHJY', 1.6783058754304387)
('B00079UM2A', 1.6783058754304387)
('B0000867GG', 1.3803117380153767)
```

Fig.[15]: Output screen of collaborative filtering

Here the results we have evaluated after many experiments considering dataset of different size and using different weight for each relation type we found the algorithm correctly identify nearest neighbors and give recommendations.

CONCLUSION

Chatbot and recommendation system was developed using techniques and theory learnt in course CPSC 8650. Collaborative filtering and clustering methods were used to provide recommendation to users by mining trends of user choices based on “bought” and “viewed” items. To interact with recommendation system, we developed a natural language processing Chatbot which recommends products, trained on user reviews of customers. We used recurrent neural network model to train on amazon review data and movie dialogue corpus. The Chatbot is an intuitive medium of conversation for users to seek suggestions on particular product.

DATASET LINKS

<https://drive.google.com/drive/folders/0ByF97F7OFu5IWGdhTG05WWJoMzQ?usp=sharing>

<https://drive.google.com/drive/folders/0ByF97F7OFu5IUkhjZE5lVxhOTA?usp=sharing>

REFERENCES

1. Amazon dataset: <http://jmcauley.ucsd.edu/data/amazon/links.html>
2. Amazon metadata: <http://snap.stanford.edu/data/amazon/productGraph/metadata.json.gz>
3. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
4. Olah, Christopher. "Understanding lstm networks." GITHUB blog, posted on August 27 (2015): 2015.
5. Tensorboard tutorials https://www.tensorflow.org/get_started/embedding_viz
6. Community detection lib: <http://perso.crans.org/aynaud/communities/>
7. Udacity Deep Learning Challenge, Siraj Raval: <https://youtu.be/SJDEOWLHYVo>
8. Collaborative Filtering Tutorials: <https://www.youtube.com/watch?v=KegVL-0vSQg&list=PLseNcwx1RJ4WdgtrMTXndw4B4nlf4-pgS>
9. https://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/recommender-systems_asanov.pdf
10. Image-based recommendations on styles and substitutes link
11. <http://cseweb.ucsd.edu/~jmcauley/pdfs/sigir15.pdf>
12. Inferring networks of substitutable and complementary products
13. <http://cseweb.ucsd.edu/~jmcauley/pdfs/kdd15.pdf>