HW3

1. Word Embedding

After running the compute features.py, I got pretrained word embeddings which contains the word vector for all the frequently used words. Also, I got custom word embeddings based on the movie reviews. Next, I will compare pretrained and custom word embeddings in neighborhood they capture.

1. Comparing words

The first thing I want to talk about is polysemy. Polysemy is common in English. So, the vector for a single word will be different if it means different in different context.

For example, the Acronym “lol”. In normal context, it always refers to the game “League of Legends”. So, if we use Genism to find most similar words, we could find that the words that are closely related to it are words closed to the game or some term only used in this game like “kaboom”, “coxhill”. However, “lol” always means “lot of laughters” or “laugh out loudly” when people write movie reviews. So, if we use Genism to find most similar words, words like “yay”, “hahaha”, “freaking” will be found out since these words are used to express their positive or negative emotions.

Another example about polysemy is “rate”. In the general context, “rate” always means “speed”. So, in the pretrained word embeddings, words that most similar with “rate” are “increase”, “higher”, “inflation” and “growth”. But in the context of reviewing movies, “rate” means people’s evaluation for the movie. So, according the result of “most\_similar” function, we could find that words like “foremost”, “recommend”, “rank”, “suck” are most similar with “rate” since these words show people’s evaluation for the review.

Another difference I want to talk about is that some words will be more closed to each other in some specific context. For example, if we use the “Rank” command, we could find that “bathroom” is the 8th most related to “phone” in custom word embeddings. That’s because people always talk about “phone” and “bathroom” together and they think audience are supposed to go to bathroom to answer the phone. In comparison, in the pretrianed word embeddings, “bathroom” is the 1885th related to “phone”.

Also, Pretrained word embedding tends to put names together like “Macarthur”, “Truman”, “eisenhower” etc. While custom word embedding not.

1. Comparing reviews

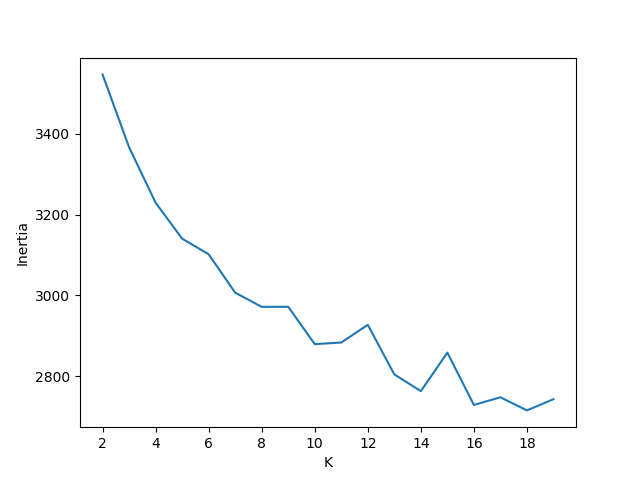
After converting each review into a vector using pretrained, custom word embeddings and TfidfCounvectorizer, then I used cosine distance to find 4 nearest neighbors for a particular review based on these 3 different types of review representation.

Firstly, we could found that word embedding methods work much better than using sparse vector to find similar reviews. For example, the one that is most similar with target review got distance 0.632 using sparse vector. It is on some degree because the sparse vector contains so many 0 in it. However, the distance between nearest review and target review would be 0.0043 if we use word embeddings to convert reviews into vectors and calculate cosine distance.

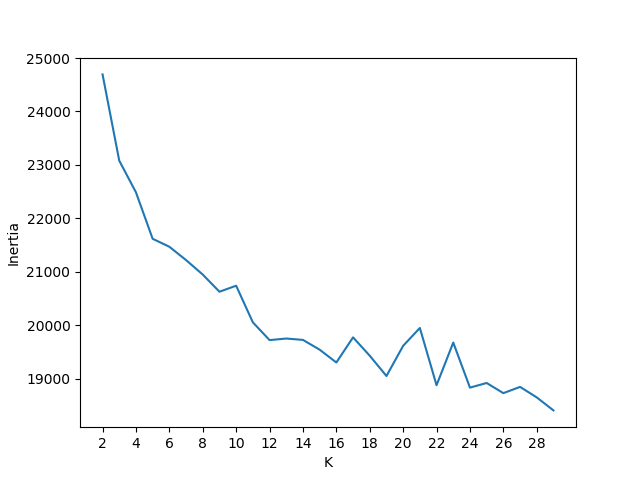
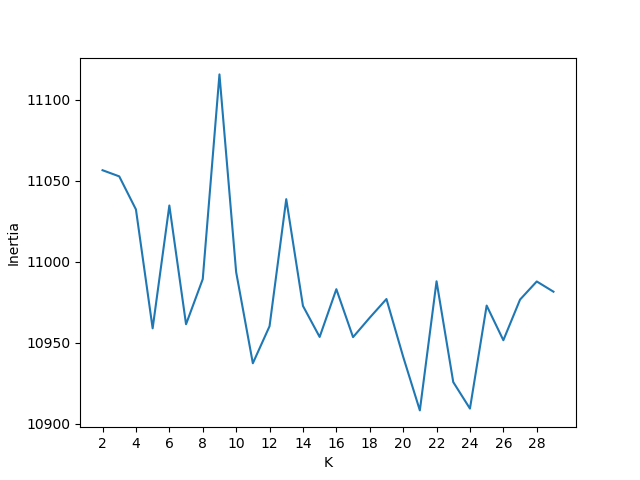
Another point I want to talk about is that if we use sparse vector to calculate distance and find nearest neighbor, it would give you the one that contain exactly same frequent words with target reviews. For example, the one that was selected as nearest neighbor contain lots of frequent words like “show”, “enjoy” which also appear very frequently in target review. However, target review is labeled as “negative” but this neighbor is labeled as “positive” which is a wrong neighbor actually.

For word embedding, it’s another advantage over sparse vector is that it could found out lots of synonyms based on the context of each word. For example, the target reviews contain a phrase “low budget”. Then the most similar review based on pretrained word embeddings contains “a budget of five shillings and sixpence”. So, we could see that word embedding methods could help to find out those synonym and increase the recall score.

Another difference between pretrained word embedding and custom word embedding is that sometimes custom word embedding is not that good at finding neighbor reviews with same sentiment. For example, for the review with idx=150, all the neighbor reviews found by custom word embeddings have opposite sentiment with target review. That’s because it treats “best” and “worst” as similar words since people always say these words in similar sentence structure when making movie reviews.

1. Clustering
2.  Picking K

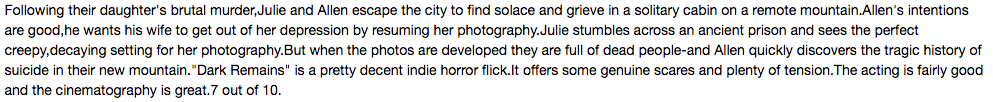
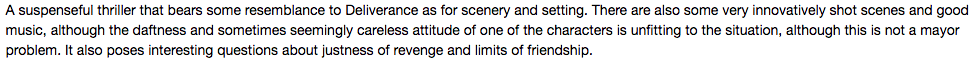
For pretrianed vectors, the K I pick is 10. Since it is the “elbow point” and the line becomes flat after 10.



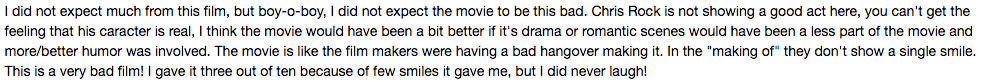
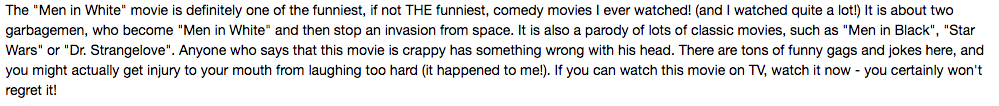
For custom vectors, the K I pick is the 12 since the line decreasing rate of the line drops a lot after 12.

For sparse vectors, the K I pick is 5. After 5 this line increases first and keep fluctuating.

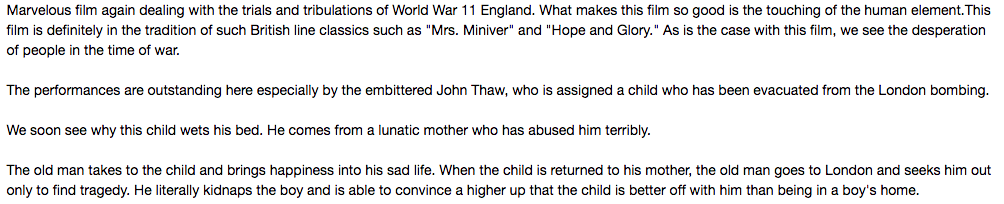
1. Comparing Clusters

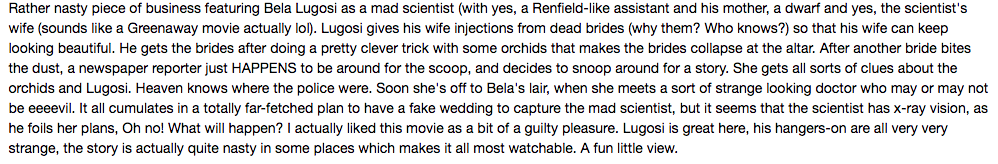
For Pretrained word embedding methods, the first cluster I selected is the one with idx=4.

After reading reviews under this cluster, I also did word frequency analysis for this cluster, I found that words like “murder”, “escape”, “suspenseful”, “dead” are more frequently mentioned than other words. So, combined with the reviews I read, I want to tag this cluster with “reviews for suspense movies”.

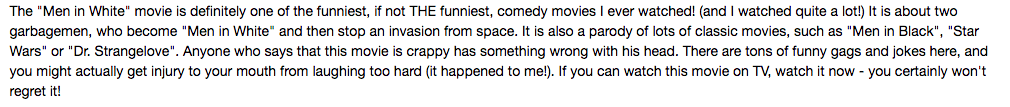
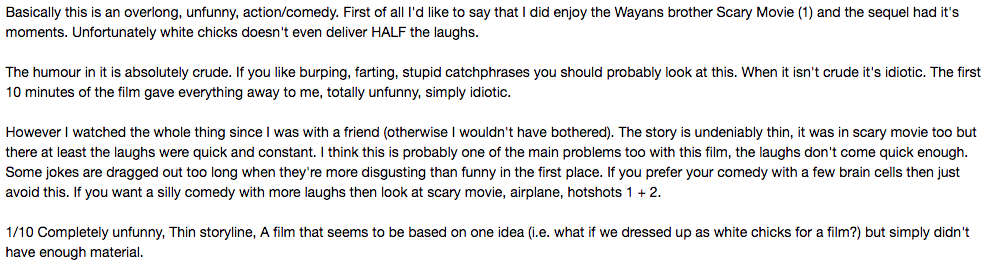
The second cluster I selected is the one with idx=8

From reviews under this cluster, we could clearly see that most reviews are about comedy no matter people’s review are positive or negative. Words like “drama”, “humor”, “laugh” and “funny” appear frequently.

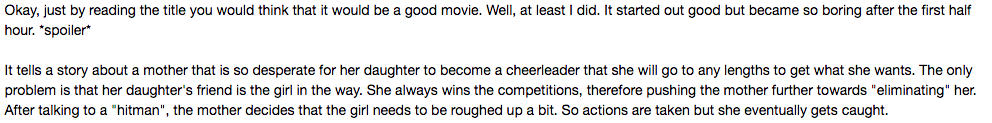
For custom word embedding methods, the first cluster I selected is the one with idx=11

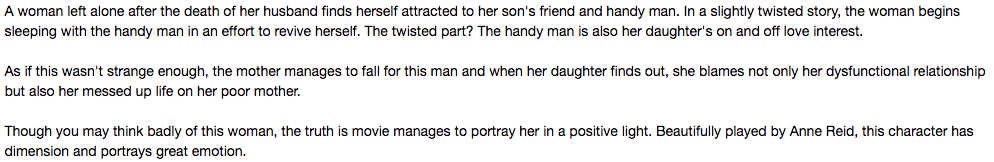


Under this cluster, I also did a word frequency analysis, “macarthur” even appears 9 times. After reading reviews under this cluster, I found that words like “war”, “bomb”, “police”, “guilty” also appear many times. So, I want to tag it with “reviews about war and crime movies”

The second cluster I selected is the one with idx=9

Firstly, according to the overlap analysis, this cluster is very similar with cluster #8 under pretrained method. After reading reviews, I found that lots of reviews are also about funny. So I want to tag it as “reviews about comedies”.

For Sparse vector methods, the first cluster I selected is the one with idx= 1

Under this cluster, I found that many reviews are related to different roles in family like “mother”, “father”, “husband”, “wife” and “daughter” etc. So, I’d like to tag this one as “reviews about family drama”

Screen%20Shot%202018-05-31%20at%203.53.26%20AM.pngAnother cluster I selected is idx=3

Screen%20Shot%202018-05-31%20at%203.52.51%20AM.pngFor this cluster, I didn’t find that the reviews are about one specific topic. However, I found that nearly all the reviews under this cluster express clearly whether they recommend this movie and show their clear positive or negative attitude about the movie. Additionally, reviews under this cluster don’t have many descriptions about plots.

As for the differences between these 3 methods, the feature of pretrained word embedding method is that words in it are trained by a very large and general dataset, which is not that related to movies. For example, the feature of cluster #7 under pretrained methods is that all the reviews contain many people names since names under pretrained methods are similar; for custom word embedding method, one feature is that it always put negative words and positive words as similar words so that reviews under one cluster are always about movies with same themes while not necessary same sentiment like cluster #9 and #11; For sparse vector method, the feature of it is that it always capture exactly same common words in different movies and put them in the same cluster.

1. Purity

The purity for pretrained vector is 62.26%; the purity for custom vector is 62.60%; the Purity for sparse vector is 58.92%. they do not match my expectation since the purity is kind of low. However, it is acceptable since there is a trade-off. For example, if we set each review as a cluster, then the purity would be 100%. But this would not helpful at all and it cannot summarize information for us.

1. Collaboration

I mainly discussed with Gloria Gong and Lingyun Guan.

My computer got bug firstly and I cannot run the code. Gloria also got similar bug. So, I asked Gloria Gong and we figured it out together by searching on google and found solution to debug it. Additionally, we also discussed the logic behind pretrained word embedding, custom word embedding and sparse vectors as well as how the distance was calculated to find the neareast neighbor and make cluster. After that, we got better understanding about the algorithm which is good for us to analyze the result.

With Lingyun Guan, we discussed some ideas to compare words and reviews between 2 sets of embedding and got better understanding about pretrained and custom embeddings.