Recommendation of Similar Articles

Authors:

Alec Jackson

Abstract

Cosine similarity is a common technique in language processing to establish how closely related two vectors are. This is one of the more common approaches to recommendation systems which operate under the assumption, two more closely related vectors are more suited to a user's preferences than two further vectors. The aim of this implementation was to attempt to an alternative method to cosine similarity that accounted for variation in word choice for an article recommendation system.

Introduction

Recommendation systems of all kinds have been becoming increasingly important with an ever growing amount of data on users and content. News recommendation has been increasingly becoming a business model for many large companies seeking to attract and maintain users. These systems need to be efficient while also returning quality matches for an end user. However as privacy becomes an increasing level on concern to users, companies have been seeking to gain users trust by collecting and using less data when possible. One of the common approaches to recommendation systems is to recommend things that similar users have liked. The problem is this approach uses a lot of personal information to generate relevant results.

This System is meant to rely only on a corpus that the user has created to represent their interests while potentially being efficient enough to run entirely client side. This would have the advantage of never transmitting any user data to servers. Relying on a corpus instead of user data makes this problem more complicated. More advanced classification systems will need to be used to make this technique competitive with the results that can be produced from a similar user approach.

Related Work

Recommendation systems are a fairly well documented topic in natural language processing. There appears to be two main approaches to this kind of system. The first is a user similarity approach, where through some kind of metric, either interests, demographics, or something else a vector is created to represent a user. Then the stuff recommended to that user is stuff that vectors, or users with high similarity to that user, have all been viewed or liked. This system relies on a large amount of data about both users and what they do. The other system is more naive,

however more advanced version of it also exist. The idea behind this system is to convert content into a vector and compare that vector to other content. Things similar to that content are then recommended to a user. This approach is what my model is based off of.

System Description

This system uses a cosine similarity approach, but has a few extra ideas and objectives. This model uses an implementation of Word2Vec to train the system on how some words are related to others. This is meant to take into account how language can vary. In a basic cosine similarity model no consideration is made for how words relate to each other. In this model the idea is to extract the most similar words and use those in addition to the actual word to determine similarity.

Data

This system is going to use three data-sets. The first represents the largest set. This is the training set for the Word2Vec model. This is going to be what decides how words in the model relate to each other. It is important that this model is both large and reflects a close approximation of what the other data looks like. Training this model on anything too specific will result in bad similarity matches. This especially applies to homographs, words that are spelled the same but mean different things, for instance, if the model is exclusive on information about semiconductors and computers the similar words to chips will more likely reflect computer chips than something like potato chips which has the potential to create false positives, or false negatives, in the data.

The second set of data is going to represent a user's interests. The interests are going to go through some processing to remove stop words and punctuation. These tokens are usually not important to the overall meaning of a document so removing them helps to isolate the important parts of the corpus. After this the set will need to be further processed in a few ways. A raw word count should be made, this will allow for a naive implementation of cosine similarity. Second, a list of the frequent words in the corpus should be created. As long as frequent words are not generic they are an important part of the data. The third thing that should be created is a list of the most important words in the set, using term frequency inverse document frequency scores (TF/IDF). Frequent words, and important words will work together to represent the most important words in the corpus.

The third data-set is the input set. This is what needs to be processed by the model and classified as a match or not. The processing on this set is the most basic. The same kind of stop word and punctuation removal needs to occur for

the same reasons as in the training set. Then the top few important words should be extracted.

Algorithm

The algorithm used for this implementation mostly does the processing with a keyword approach. After all the data has been preprocessed the top ten similar words are extracted from each of the following important words from the article. This is done with the Word2Vec model that was trained on the corpus of news. The words that are returned are the words that appear the most similar situations to the input words. These words are then checked against the frequent and important words in the training set. Each match that occurs gets recorded as such and adds a point to the number of matches that have occurred. Depending on where this threshold is set then an article which has enough matches will be run through a basic cosine similarity function. Since the Word2Vec performs poorly on short articles because the words that get extracted as important are too generic the cosine similarity model is helpful. This performs much better on small data sets, small datasets will usually score low in the cosine similarity model for being too generic.

The purpose of the Word2Vec approach is to be able to handle rather small datasets. This is supposed to reflect the real world applications of a system like this. For this kind of system the model may not have very much data from a user at first. This models starts returning relevant matches with relatively small input data.

Experiment

All of the experimental results for this were setup using the same datasets. A large corpus of news was used to simulate the real world application of news on different topics from different news sources. Since all the experiments were run off of the same dataset there and the training data did not come from the same dataset there is concerns about the quality of the results.

Results

The articles that got returned by the algorithm were mostly similar to the topics it was trained on. In a relatively subjective analysis of the results, when the model was trained on “tech news” news articles relating to computer software and hardware companies 80% of the first 25 articles returned were about computer software or hardware. When trained on news articles relating to sports, the number was much lower only 40% of articles being about sports. Although this number does get a bit higher if articles about sports stars or other celebrities are factored in since the model seems to have a difficult time determining the difference between those. The data about why the model selected what it did provides some insights into why the model did so much better on the tech news than the sports

news. In sports news whenever a team was mentioned in the article, or the animal or thing the team is named after it usually is identified as a similar word than then recognizes the other sports teams as a similar item. Additionally many sports terms and political terms appear to be shared or used very similarly. Words like addressing would be seen as similar to tackle which would make the system recognize an article as similar. Sports articles often mentioned players being in “talks” with a team or the team “discussing” player salaries or trades. These words also appeared as important words in political articles referring to trade talks or discussions with countries.

Tech Headline (First 10)	Subjective Match or No Match (M or N)
How to watch the Google I/O keynote live	M
Former Microsoft executive Doug Burgum is North Dakota’s next governor	M
Huawei says shipped 100 million smartphones this year as of end-May	M
Twitter isn’t hosting its annual developers conference, Flight, this year	M
Apple shouldn't worry about Google's $1.1B HTC deal just yet	M
Apple education event: How to watch live, start time	M
Facebook CEO Mark Zuckerberg says the company has a ‘responsibility to protect your data’	M
Global internet speeds got 30 percent faster in 2017	M
Facebook’s new ads will track which stores you visit	M
Brazil's Petrobras prices secondary offering at 30.25 reais per share -sources	N
Sports Headlines	Subject Match or No Match (M or N)
Macron to discuss Iran with Merkel and May ahead of Trump decision: Elysee	N
China-U.S. trade talks 'making a final sprint': state media	N

Blake Griffin is going to stay with the Clippers	M
Republican Flake plans to vote 'yes' for Kavanaugh confirmation: MSNBC	N
Obama picks Merrick Garland for Supreme Court	N
Twilio is hosting a live coding session on the New York Stock Exchange floor to watch its IPO open high	N
Shaun White wins men's half-pipe Olympic gold medal in dramatic finish	M
Nike probably doesn’t mind if angry white dudes burn their shoes	M
Mark Wahlberg Cashes In On Cleveland Browns Wins Bet, Proves He's a Genius	M
Michael Bloomberg hasn't given up hope on Trump	N

Accuracy of classification is only one was to test this though. Future tests on this system to test other things would be good. One good measurement would be how many articles it misses. Testing this on a corpus of just one type of news and seeing how many of it returns as matches would be interesting. However, an objective measurement of this kind of system may not be completely possible. Exactly what is relevant is subjective. Doing an 80 – 20 split on a dataset of one kind of news could be a good metric on how to benchmark this systems performance.

Conclusion

This approach provided compelling results that were limited by a few major setbacks. The model appeared to work for tech news sources, but struggled when run on sports news because of homographs and common important words being used across different topics. Although this does not mean this approach is fundamentally unworkable it means additional caution and care will have to be taken to produce consistently good results.

There is plenty of future work I would like to do on this project. There are a few, both small and large, tweaks to be made to the model. There are also a few larger projects like transforming this

into a more usable form. Currently it is setup to take input from a specific csv file. I would like to eventually expand this into being able to read data from RSS feeds to determine if a user would like a given article off of that feed. Furthermore, one of the current issues is under selection. It is possible this is resulting from the training sets being very specific and the news corpus focusing more on mainstream news. However a .5% selection rate is still very low from this. I would like to be able to improve this rate without effecting the rate of correct articles. One of the ways I think this could be done would be by expanding the stop word list, or by in addition to the stop word list filtering out very common words. There is plenty of room to keep working on this project, but the model stands as a proof of concept for this kind of article recommendation system.