Cross Culture Analysis via Media News Videos

Kathleen Lee, Supervisor: John. R. Kender

Columbia University

Introduction

Online video-sharing platforms, such as YouTube, are increasingly allowing the cross-border flow of media content. More specifically, these media-sharing websites are becoming an evolving news medium as they growingly share more diverse international news on their platforms. As a global media-sharing platform, websites like YouTube allows both professionals and amateur producers to participate in the content production of the news videos. In this regard, these platforms provide various insights into the heterogeneous audience group, revealing their cultural values and openness, as they share and comment on various news topics. Using this trend of consumption and production of news videos on media-sharing platforms, this study aims to examine the difference in cultural responses to particular events by extracting visual and semantic features from news videos.

Previous research on this subject have concluded that ‘AlphaGo’ is a topic covered internationally in many media-sharing websites. AlphaGo’s 4-1 victory in Seoul, South Korea, in March 2016, was in fact watched by over 200 million people worldwide [1]. With this in mind, we chose ‘AlphaGo’ as the topic of study to analyze the cultural difference toward the subject between the United States and China. YouTube was used to retrieve news videos covering ‘AlphaGo’ in the United States since it is the most popular platform for media consumption on the web in the US, with more than one billion viewers every day. Unlike in the US, there is no one singular online video platform widely-used in China, so news videos were collected from

various sources including and not limited to Bilibili, Iqiyi, Tencent Videos, and Youku. Audio was extracted from these collected videos and transcribed into transcript text using Natural Language Processing pipelines and libraries available in Python.

Some of the problems that I improved upon from the previous work were filtering search queries and implementing better categorization of words within the transcript. The previous search queries for 'AlphaGo' videos failed to remove advertisement and irrelevant videos. As a solution, I aimed to create a list of whitelisted words that would indicate whether a video was relevant or not. These whitelisted words were generated by extracting keywords from the transcripts. Finding keywords would also better categorize words within the transcript when generating image-text pairs. Depending on the word distribution in the transcript, for a news video that runs for 5 minutes, there were about an average of 300 image-text pairs available. We wanted to narrow down the words to a few keywords in order to extract images based on those keywords to highlight any cultural differences between the two countries video sources.

Related Work

In my initial research to find a way to extract only the key content within a video, I came across a paper produced by Carnegie Mellon University School of Computer Science which discusses a method for 'Video Skimming for Quick Browsing based on Audio and Image Characterization' [2]. In the paper, it discusses how the extraction of significant information is made possible through the language processing of the transcript. In particular, the method of language analysis they focused on is the technique known as TF-IDF (Term Frequency Inverse Document Frequency) to identify keywords and their relative important in the video source. As the name suggests, a high TF-IDF score indicates the importance of the word in the video. Words

that frequently appear in the transcript but are rare in the corpus are given the highest weights and categorized as a ‘keyword’ in the transcript.

To test whether this condition could be applied to our study, I found the most important words in the transcripts by calculating the total frequency of each word. An example of the result is shown in Figure 1.

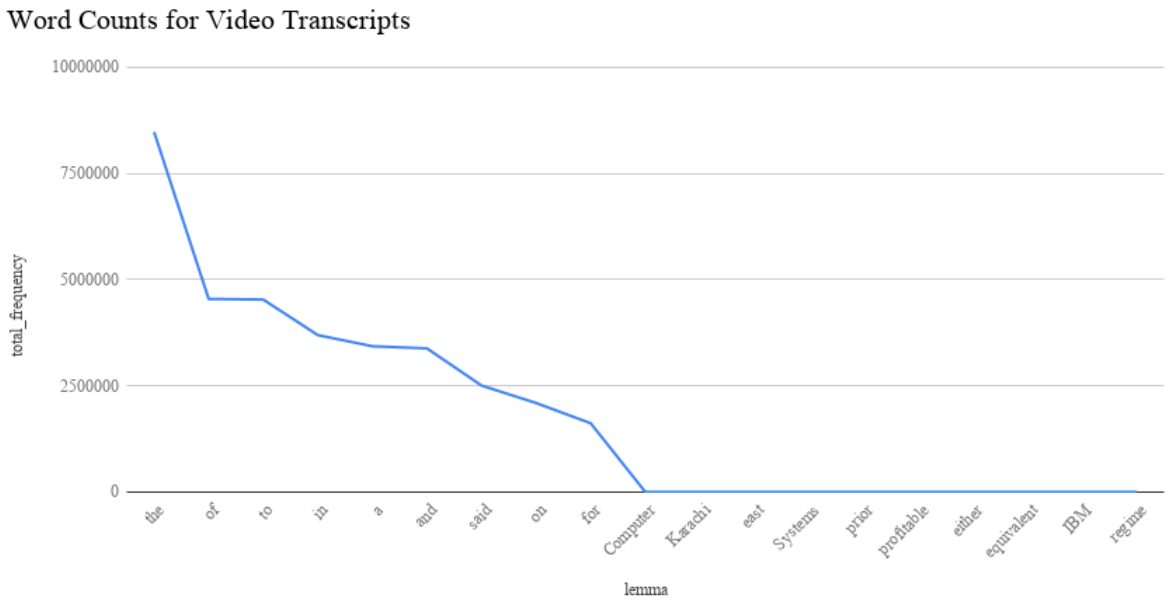


Figure 1: Sample of Total Frequency of Words from Transcripts from All Video Sources Collected

As Figure 1 shows, the words that appear most frequently in the transcripts were determinants and prepositions. Nouns such as company and people names, which may be more significant for our analysis, had a lower frequency count. Looking at this result, we concluded that TF-IDF may be technique we can use to determine keywords in our transcripts.

Data Preparation

Reuters Corpus

To have an accurate depiction of keywords used in news articles, I aimed to collect a corpus with a large set of news articles. My first attempt was to use the Reuters Corpus Volume I (RCV I), which is an archive of over 800,000 manually categorized newswire stories made available by Reuters, Ltd. [3]. Reuters is the largest international text and television news agency, which has an editorial division that produces more than 11,000 stories in 23 languages a day. I believed that this corpus would be a great source to get a substantial amount of news articles in multi-languages. The RCV I archive consists of all and only English language stories that were published between August 20, 1996 and August 19, 1997 [3]. I believed it was appropriate for our study since it contains content that ranges from English language international newswire to economic and political stories. The original dataset was formatted in XML which I converted to TXT when reading through the files.

One problem I encountered after extracting keywords from the video transcripts using the Reuters corpus was that the word ‘AlphaGo’ did not appear in the keywords. Below is one transcript example from which I got keywords:

> South Korea top class go player isidora continues go strong matched Google AI computer alphago Korean Maxim cop Lee defeated fellow Dent Pro Song June two wins take cut total recordbreaking five wins Korean go Grand Masters championships creative tactical moves reminiscent previous landlord match alphago Rayleigh seem well within reach regaining top go players seat Curry last ring never domestically November

Figure 2 shows the keyword result for this transcript. As you can see, although the word 'alphago' appears in the transcript, it does not appear in the keyword result.

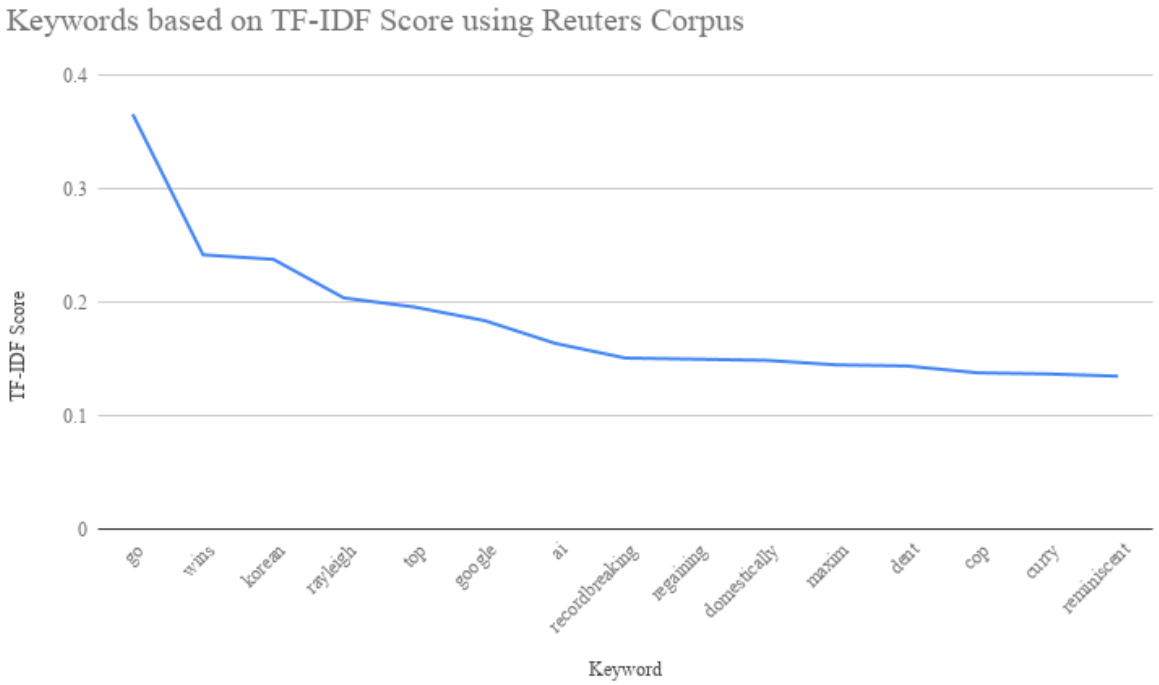


Figure 2: Keywords Based on TF-IDF Score Using Reuters Corpus

I concluded this was due to the fact that the term 'alphago' did not exist during 1996 and 1997. Thus, the frequency of the word 'alphago' in the corpus was zero and the TF-IDF score was undefined. A solution to this problem was to add or change the corpus to a more recent set of news articles. Specifically, a corpus set that contains news articles published beyond 2016.

New York Times Corpus

A more recent corpus of news articles I found was The New York Times Annotated Corpus compiled by the Linguistic Data Consortium at the University of Pennsylvania [4]. The New York Times Annotated Corpus contains over 1.8 million articles written and published by the New York Times between January 1, 1987 and June 19, 2007 [4]. This corpus was not readily

available online so I gained access to it through the Natural Language Processing lab at Columbia University under the supervision of Professor Julia Hirschberg. The original data was formatted in XML which I converted to TXT when reading through the files. Although this corpus contained more recent articles published than Reuters, it still did not contain articles published after 2016. Therefore, I decided it was necessary to collect more articles from the New York Times published within the last five years.

News Scraping from the New York Times

News scraping is the process of extracting data, in particular news articles, from the web. To automate this process of downloading articles from the New York Times, I used the Python package BeautifulSoup. BeautifulSoup parses the HTML/CSS components of the individual web page and uses the hyperlink to select specific components of the web page. I extracted news articles from https://spiderbites.nytimes.com/, which is an archive of free news articles published by the New York Times organized by year and month as shown in Figure 3 and 4. I mainly focused on collecting articles published within 2014 to 2019.

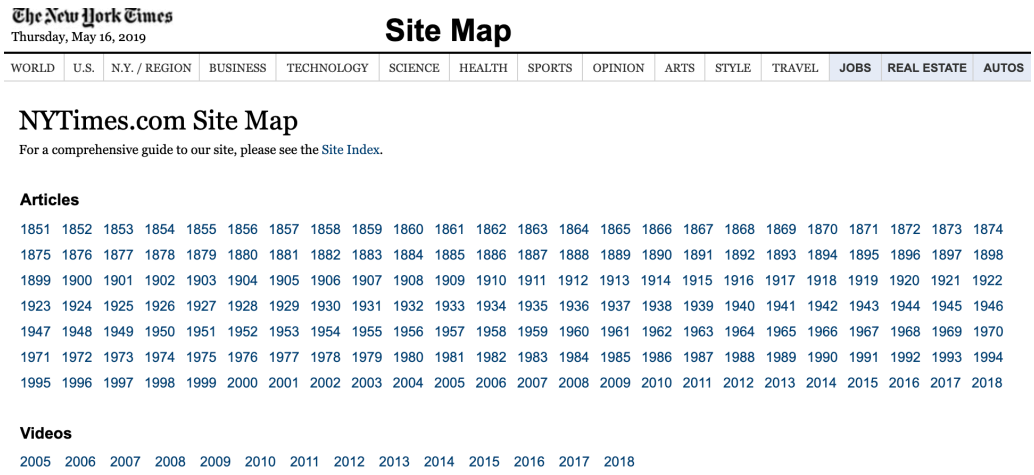


Figure 3: Collection of New York Times Articles Organized by Year

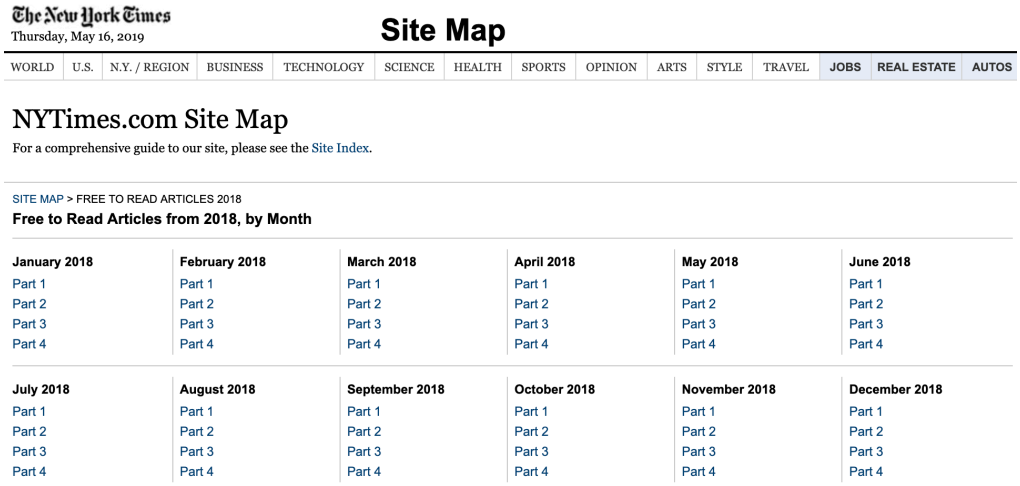


Figure 4: Collection of New York Times Articles Published in 2018 Organized by Month

Articles within each month were divided into four parts. I created a Python script such that it iterates through each of the parts and collects the links to the article pages. I achieved this by storing all article reference links that has the `<a href= “>` tag that were within the `<ul id=”headlines”>` div, as shown in the highlighted section of Figure 5 below.

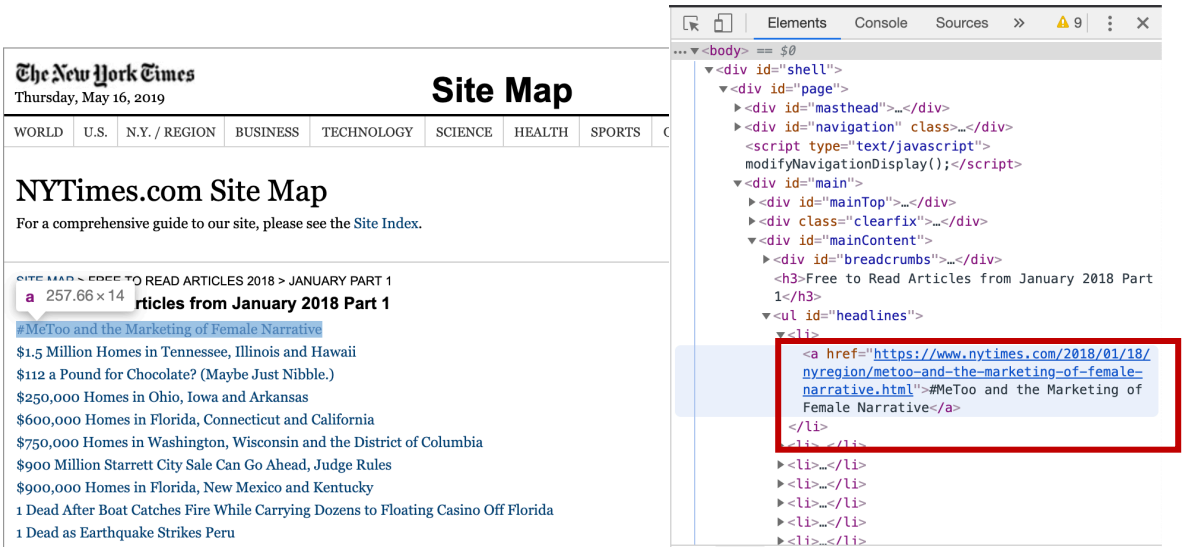


Figure 5: Collection of New York Times Articles Published in January 2018 (the First of Four Parts) and HTML page

To find the exact HTML class that includes the title, body paragraph, and advertisements of the articles, I manually examined each of the article's HTML source page. From there, I noticed that articles titles were given the class 'headline' as shown in Figure 6. Like this, I individually parsed the HTML classes and found that the dates were given class 'css-1w5cs23 epjyd6m2', the body paragraph given ''css-1i2y565', and lastly advertisements had the class ''css-1soubk3 epkadsg3'. This method allowed me to filter the articles before downloading them by removing all classes that contained advertisement words. I then formatted the article data into XML file. Using this method, I collected over 200,000 additional news articles from the New York Times to add to the corpus.

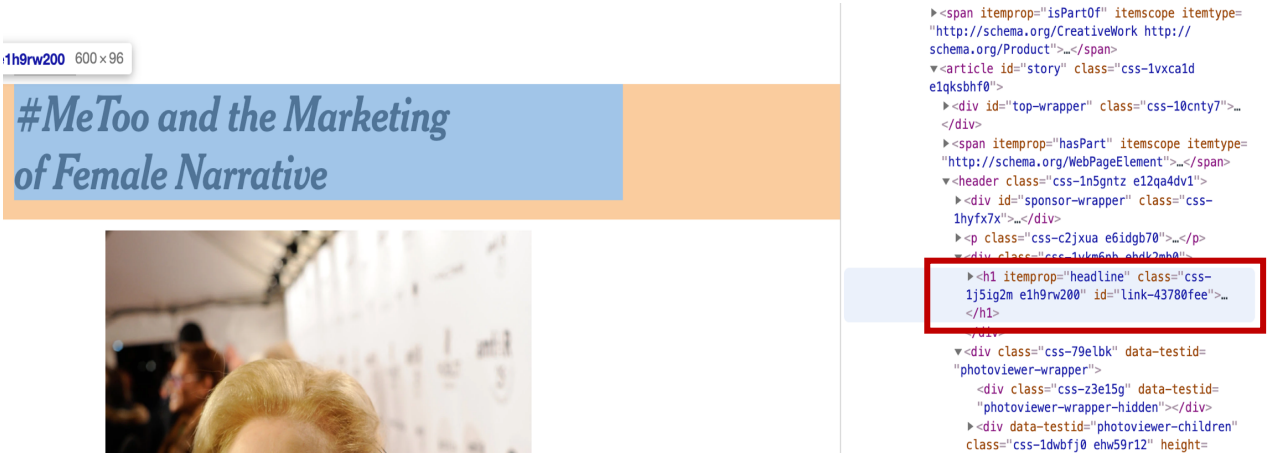


Figure 6: HTML Page for a New York Times Article Showing Title Class

Implementation Details

Data Pre-processing

In order to pre-process the newspaper article data, I used several Natural Language Tool Kit (NLTK) libraries, which is a Python platform available to use for natural language processing, and the sklearn library, which is an open-source machine learning tool available in Python. First, I converted all the newspaper articles to TXT files. Since all of the New York

Times articles were in XML format, I had to parse the file to obtain the <title> <date> <body> <p> blocks of strings. I then saved the strings to a TXT file so I can access it again later without needing to parse the XML file again. Next, I removed stopwords from the data. Stopwords are common words used in sentences which usually include a large number of prepositions, pronouns, conjugations, and many others. Stopwords can also include blacklisted words, such as words related to advertisements. I believed that removing stopwords are necessary in order to focus on words that were most relevant to the context, rather than on prepositions. Checking for the most commonly used words in the transcripts, I found that the following should be added to the stopwords: `{the, of, a, and, to, in, for, that, in, we, on, with, as, this}`. Additionally, I added several advertisement words I found in the description section of the videos, such as `['http', 'https', 'com', 'wired', 'tumblr', 'facebook', 'instagram', 'www', 'youtube','weibo','twitter','wiki']`. Furthermore, I imported stopwords from the nltk.corpus library.

Next, I pre-processed the data by removing redundant text-components and normalizing the transcripts. Redundant text-components include punctuations, URL links, and stopwords. Normalizing data includes two parts: stemming, which cleans the text by removing suffixes, and lemmatization, which finds the root of the word. First, I removed punctuations from the words and converted them all to lower case. I then removed tags such as `"</?.*?>"` and removed digits and special characters like `'\W',` which are words following the character "`\`" like in a URL. I then used PorterStemmer() and WordNetLemmatizer() to get the base form of the word. The functions return the input word unchanged if it cannot be found in WordNet to prevent any incorrect or misspelled root words. I created a word cloud shown in Figure 7 using

the word cloud library to visualize the most frequently used words in the corpus after pre-processing.

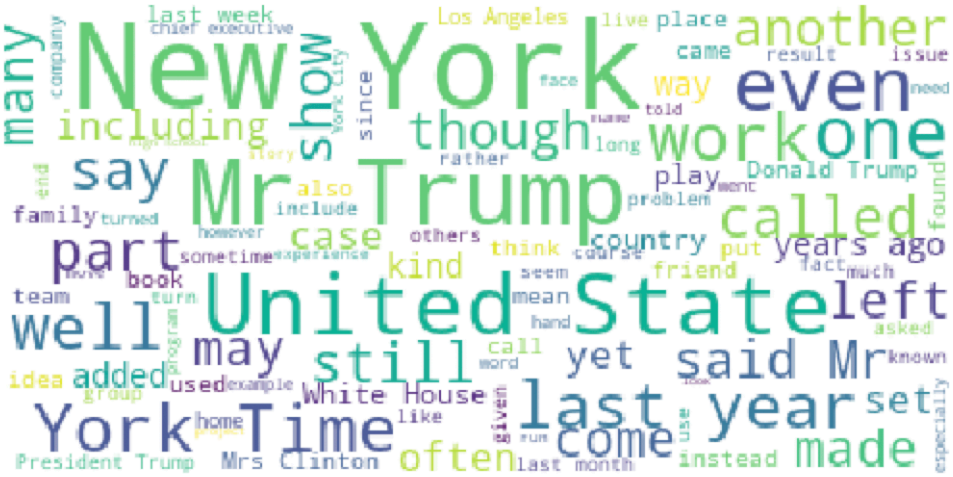


Figure 7: Word Cloud of Most Frequently Used Words in Corpus After Pre-Processing

Before running the article data through a machine learning algorithm to calculate the TF-IDF score of individual words, we needed to tokenize the long string of article text into a list of words. I used CountVectorizer() from the sklearn library to tokenize the text and convert the list of words to a matrix of integers by the process of vectorization. I created an instance of the CountVectorizer class and then used the fit_transform function to learn and build the vocabulary of known words. The parameters passed into the CountVectorizer() function are max_df, which specifies to ignore words that have a document frequency higher than the given threshold, and max_features which determines the number of columns in the matrix [5]. The functions are shown below.

```
from sklearn.feature_extraction.text import CountVectorizer
import re

cv=CountVectorizer(max_df=0.8,stop_words=STOPWORDS, max_features=10000,
ngram_range=(1,3))
word_count_vector=cv.fit_transform(corpus)
```

An encoded vector is returned with a length of the entire vocabulary. Here is a segment of 10 words in the word count vector.

```
list(cv.vocabulary_.keys())[:10]

['liverpool','ousts','manchester''united','stunning','solo','goal',

'philippe','coutinho','helped']
```

Tf-IDF Vectorizer

The next step is to refine the word counts using the TF-IDF vectorizer from sklearn. Word counts obtained using countVectorizer() contains large counts of common words rather than focusing on the context specific words in the corpus. We use the TF-IDF vectorizer to penalize words that appear frequently across the document and highlight words that are more relevant to the context. The TF-IDF consists of two parts – TF (term frequency) and IDF (Inverse document frequency), which are defined as follows:

$$TF = \frac{Frequency\ of\ the\ word\ in\ a\ document}{Total\ number\ of\ words\ in\ the\ document}$$

$$IDF = \frac{LOG(total\ documents)}{Number\ of\ documents\ that\ contain\ that\ word}$$

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer=TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_transformer.fit(word_count_vector)

# get feature names
feature_names=cv.get_feature_names()

# fetch document for which keywords needs to be extracted
doc = corpus

#generate tf-idf for the given document
tf_idf_vector=tfidf_transformer.transform(cv.transform([doc]))
```

Based on the TF-IDF scores, we can extract the words with the highest scores to obtain the keywords for that transcript. I sorted the `tf_idf_vector`, which contains the word and the corresponding tf-idf values, and then passed it through a function that extracts the top `n` words from the vector. For this research, I selected the top `n = 15` words to use as keywords.

Keywords Result

The keywords extracted for one transcript is show below. The transcript was number 16 among the collected AlphaGo videos.

```
=====Body=====well familiar chessplaying computer deep blue famously get Garry Kasparov years ago
ancient game Go known significant challenge artificial intelligence overwhelming complexity makes
intuitive game new program beat challenge huge match go project expected take place worlds long
time ago Grandmaster human alphago computer program developed Google reports development makes
milestone AIup ancient game Go product singled board game computers crack due complexity rules
simple though gain territory grid placing capturing black white stones placed almost indefinite
number combinations belief Mason change US tech giant Google says computer fiftyfifty chance beat
strongest players world quit artificial intelligence program called alphago developed Google
londonbased company deepmind Google says alphago trying learn like human observing others play
repeatedly predicting outcome game well using instincts make best move Uncharted situations gut
instinct makes alphago advanced compared previous software programs including IBM trust
supercomputer deep blue Russian chessKasparov I learn use generalpurpose algorithm interpret
games patterns Google says alphago already defeated European Potter Champion fleece five times
five tournaments October cofounder deepmind Dennis house Obby says ultimate goal alphago
development Supply solve realworld problems future I may CT scan grain image image best kind
treatment decadeslong Korean champion Potter Grandmaster isidora sold March The Winning Side
```

```
awarded million US dollars prize money whatever outcome one thing sure thecoming match mustard
development artificial intelligence news

=====Keywords=====
alphago 0.44
google 0.238
game 0.198
artificial 0.164
says 0.158
deepmind 0.152
computer 0.14
intelligence 0.134
development 0.133
grandmaster 0.133
makes 0.129
program 0.123
go 0.117
complexity 0.114
potter 0.112
```

Collecting all of the keywords found in each transcript, I compiled them into a whitelist. Figure 8 shows the top 20 keywords included in the whitelist.

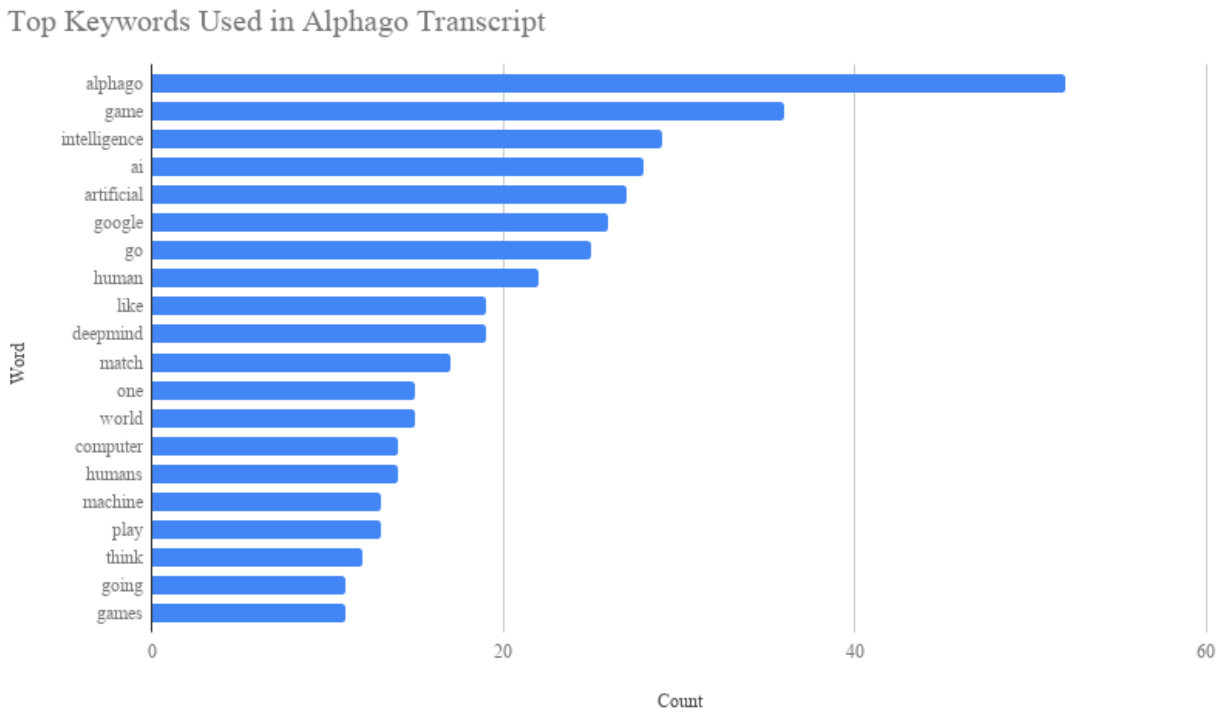


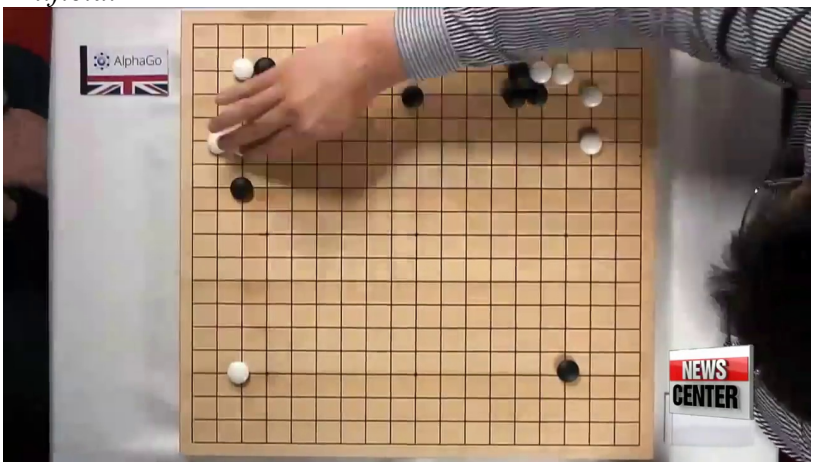
Figure 8: Top 20 Keywords Used in AlphaGo Youtube Video Transcripts

Extract Image Frames for Keywords in Video

To better understand the relationship between the keywords and the video source, I extracted images at the given timestamp of each of the keywords using an OpenCV library called VideoCapture. Through this, I wanted to see if there were any interesting alignments between keyword and their corresponding images. The images extracted from transcript number 16 are shown below. Since many of the keywords appeared multiple times in the transcript, the images were grouped together by keywords for better comparison.

Alphago

Artificial



Complexity

Computer

Deepmind

Development

Go

Intelligence

Conclusion and Future Work

Considering that the corpus had over 1.8 million newspaper articles, I could not use all of the data to extract keywords from the transcripts. I used a sample of the New York Times annotated corpus, namely articles from 2004 and 2005, along with the more recent articles that I collected through news scraping to get the TF-IDF value. One future work would be to test out different samples of the corpus to see which sample provides the best set of keywords.

An issue I noticed was that the transcript had a lot of misspelled or incorrect words, since speech-to-text algorithms are often not completely accurate. The lemmatization during pre-processing could not find the root word for these misspelled words so left them as they originally were. This cause many words to be miscounted when calculating TF-IDF scores. Another future work would be to improve upon the speech-to-text conversion of the video audio to transcript to ensure that all words are spelled correctly.

Observing the result from the image extraction, we can note that many of the frames are images of news anchors, journalists, and the game go board. Currently, the image is extracted almost instantaneously at the timestamp, perhaps with the different of $+/-$ 0.5 seconds. Future work would be to expand the timestamp to at most +/- 2 seconds to see if better connections can be made between the keywords and the images. Currently, it seems as though the keywords do not necessarily provide better association for the images. Future work could include gathering videos that have more on-site coverage of the topic rather than discussion between anchors and journalists.

References

1. “AlphaGo.” *DeepMind*, deepmind.com/research/alphago/.

2. Smith, Michael A, and Takeo Kanade. “Video Skimming for Quick Browsing Based on Audio and Image Characterization.” 30 July 1995, doi:10.1.1.33.1714&.

3. Lewis, David D, et al. “RCV1: A New Benchmark Collection for Text Categorization Research.” *Journal of Machine Learning Research 5*, 4 Apr. 2004, pp. 361–397.

4. Sandhaus, Evan. “The New York Times Annotated Corpus.” *Linguistic Data Consortium*, 17 Oct. 2008, catalog.ldc.upenn.edu/LDC2008T19.

5. Vivek, Sowmya. “Automated Keyword Extraction from Articles Using NLP.” *Medium*, Analytics Vidhya, 17 Dec. 2018, medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34.