Ezana N. Beyenne

MSDS 453, Section 57 2020

Assignment 2: **Three methods to assign numerical vectors to documents**

Abstract

This assignment takes the corpus built in the previous assignment on autonomous vehicle safety and determines the vectorization methods that do the best job of classification. I use three methods namely: (1) Analyst Judgement, (2) TF-IDF, and (3) Doc2vec to assign these numerical vectors to documents. I then used my analyst judgement to identify two classes namely: safety and technology I consider to be the subtopics of corpus. These two classes allow us to classify documents on articles about autonomous vehicle safety to be either about technology or safety in the final step of the study.

Autonomous vehicles are an inevitable part of our future, as it will hopefully reduce traffic fatalities and street congestion (Walker, 2020). The data, research and development relating to its safety are still in the early stages and because of these early stages, the consumer sentiment is divided (Alessandro, 2020). This study hopes to help management quickly identify areas of interest to either safety or technology, without having to wade the growing number of articles on autonomous vehicles.

Introduction

An increasing number of articles are being written about autonomous vehicles especially on the artificial intelligence that goes into building them. The technology involves a lot of computer vision, natural language processing algorithms, and software engineering complex in nature. The technology is expected to have a positive impact on issues ranging from the environment to road congestion.

Autonomous vehicles are expected to reduce the instance of impaired driving due to either alcohol or drug impairment. Secondly, it will help reduce emissions and costs by finding the fastest routes to destinations thereby improving fuel efficiency. According to the Department of Transportation and the National Highway Traffic Safety Administration, human error causes almost 94% of accidents on US roads (Alessandro, 2020). This study hopes to identify the vectorization method doing the best job of classifying autonomous vehicle safety articles into either being about safety or the technology subtopic.

Literature review

Regulators and companies at all levels try to keep track of the ever-evolving nature of this technology with an eye on both the technology and the safety aspect of it. The organization's need to stay abreast of the rapidly changing technological and safety landscape of autonomous vehicles is a necessity. According to Rand Corporation's research, autonomous vehicles would have to be driven hundreds of millions of miles, (sometimes hundreds of billions of miles), translating to decades, if not centuries to achieve these goals. Since this is not feasible, Rand Corporation recommends regulations that are adaptive in nature, so such regulations harness the benefits while mitigating the risks of this rapidly evolving technology (Kalra et al., 2016). There are currently thirty-seven states, along with the District of Columbia, that have created legislation or issued executive orders about autonomous vehicles.

Methods

In this study, I used three different methods for assigning numerical vectors to represent each document found in the corpus. I then created a documents-by-term matrix for each vectorized method. After some testing I found that individual words (terms) got better than using bi-grams or n-grams. The first step was preprocessing the documents by removing unnecessary

tags, punctuations, images. I then tried to apply a stemmer and compared that with a lemmatizer and I found that the lemmatizer had better results. The lemmatizer is like the stemmer, which the difference being that it can capture canonical forms based on the word's lemma. Once the preprocessing was out of the way, I worked on the three vectorization methods are as follows:

Approach 1: *Analyst Judgement*

In this section, I used corpus wide statistics to identify important terms. To decide the importance of a term found in the corpus, it must have the following characteristics: (1) it is important in at least one document, and (2) it is prevalent in multiple documents. I used sklearn's CountVectorizer with a max feature length of one hundred to give me the document vectors and derive the top six most frequent and the six least important terms. I then used this information to create two classes "safety" and "technology." Next, I manually derived the subtopics that I believed applied to either the "safety" or "technology" classes, by reading a sampling of the articles. Whenever I read an article that contained safety, crash, people, policy, testing etc. the article discussed "safety", as it discussed lidar, technology, research, system etc., factors involving "technology". Table 1 shows the class terms and their manually derived equivalent terms.

Table 1: Two classes and their manually derived equivalent terms

	Manually Derived Equivalent Terms									
Class Terms										
safety	safety	crash	people	driver	human	policy	standard	rule	government	testing
technology	technology	lidar	autonomous	research	selfdriving	electric	engineer	system		

I then used this information to create a function that determined the labels namely "safety" or "technology". I counted the occurrences of the manually derived terms for each class,

and if there was more safety related information, the label would be a zero, otherwise a one for technology.

Approach 2: *TF-IDF*

I used sklearn's TfidfVectorizer with a max feature length of one hundred to fit and transform the preprocessed documents to create the document vectors. As with the CountVectorizer, I found that using a word instead of a bi-gram, or n-gram produced better results. I then captured the six most frequent terms and six least frequent terms according to the TfidfVectorizer algorithm. The top six most important terms matched the ones from approach one, with the only different being the order of the last 2 terms. The least six terms did not match the output from approach one.

Approach 3: *Doc2Vec*

I used gensim's Doc2Vec vectorizer with a vector size of one hundred, and workers count set to sixteen CPUs to also derive the top six most and least important terms. I then compared the results with the other vectorization algorithms to find common terms amount them to either validate or modify my class terms. The Doc2Vec algorithm's top six important terms matched exactly with approach one, but differed with approach two on the order of the last two terms.

The six most important terms that the three vectorization algorithms determined are the most important are shown in Table 2. They matched the names of the terms, with the only difference being on the order of the terms safety and technology. The CountVectorizer and Doc2Vec had safety as the fifth most important term and technology as the sixth most important terms, whereas the TfidfVectorizer flipped that around.

Table 2: Top six most important terms for each algorithm

A1-Freq		TFIDF-Freq		A3-Doc2Vec-Count	
Terms		Terms		Terms	
vehicle	11.273263	vehicle	0.238004	vehicle	9571
selfdriving	6.707892	selfdriving	0.161158	selfdriving	5695
autonomous	5.518257	autonomous	0.147179	autonomous	4685
company	5.103651	company	0.140209	company	4333
safety	4.186101	technology	0.104772	safety	3554
technology	4.121319	safety	0.104280	technology	3499

The really big difference was in the least important six terms. The only term that was found in two of the three algorithms was “rule”. The CountVectorizer and TfidfVectorizer algorithms both had that terms, but not in the same order of relevance as seen in the Table 3.

Table 3: Bottom six most least terms for each algorithm

A1-Freq		TFIDF-Freq		A3-Doc2Vec-Count	
Terms		Terms		Terms	
engineer	0.620730	question	0.024738	star	2
transit	0.619552	rule	0.024702	telemetry	2
policy	0.619552	government	0.024479	canceling	2
electric	0.613663	driven	0.023059	techified	2
better	0.612485	standard	0.022549	accordingly	2
rule	0.604240	policy	0.020624	innovating	2

Random Forest Classification with Manual Labeling

The next part of the research involved preparing the numerical matrices for further analysis. First, I associated each row in the matrix with an eight-character string representing the document name and the columns were the terms found in the document. I split the corpus into an eighty percent training and twenty percent testing. I was then able to use a Random Forest classification algorithm to determine the vectorization method that did the best job of classifying documents into a “safety” or “technology” subtopic. The Tf-IDF vectorization algorithm ended up with the best F1 score, followed by the CountVectorization, and the Doc2vec with fifty

vectors as shown in Table 4.

Table 4: F1 classification scores of the vectorization algorithms with manual labeling.

Algorithm	F1 classificaton performance in test set with manual labeling
TF-IDF/Random forest classification	0.981
CountVec/Random forest classification	0.956
Doc2Vec_50/Random forest classification	0.956
Doc2Vec_100/Random forest classification	0.943
Doc2Vec_200/Random forest classification	0.943

Random Forest Classification with K-Means Labeling

The process of manually generating the labels proved to be a very labor-intensive task. So, I wanted to know if I could use the K-means algorithm to generate the labels for each of the three vectorization algorithms. I would first create the vectors from each of the algorithms that was trained on the whole corpus, then to run it through the k-means algorithm. The k-means algorithm then generated the cluster labels that I used as the Y variable. I then did a train test split with the vector and cluster labels. I then ran that through the Random forest classifier to see which vectorization algorithm produced the best F1 score as shown in Table 5.

Table 5: F1 classification scores of the vectorization algorithms with k-means labeling.

Algorithm	F1 classificaton performance in test set with k-means labeling
TF-IDF/Random forest classification	1.000
Doc2Vec_50/Random forest classification	0.994
Doc2Vec_100/Random forest classification	0.994
Doc2Vec_200/Random forest classification	0.994
CountVec/Random forest classification	0.949

Results

When I created the training/test set based on the manual labeling, my F1 scores were all perfect, which led me to believe that I was overfitting the data. The solution, was to use random

sampling when doing the train/test split. The result was that I have high F1 accuracy scores, with Tf-IDF having the best F1 score.

When using the k-means clustering algorithm to generate the labels, I experimented with setting the number of clusters from one to five, and settled on three, since it did not seem to generate classifications with perfect F1 scores. The Tf-IDF algorithms again had the best score. The difference in the F1 classification scores, between the two methods of generating labels, is that the Doc2Vec algorithms performed much better when using the k-means cluster. Additionally, the classification scores overall were better when using the k-means generated labels.

Conclusions

Autonomous vehicle technology is not a matter of if it happens, but when. This technology would evolve faster if safety is adopted and assessed at all stages. There are a lot of articles suggesting that safety is at the primary concern from developers, to the consumers, and to the regulators. This research assignment's ability to classify the articles on autonomous vehicle safety into either "technology" or "safety" related subtopics would help the organization's various departments focus in on their interests quickly, without having to wade through ever growing number of articles that are out there on the topic. For example, as the legal department will focus on safety laws, research department would focus on both the technology and safety.

Works Cited

Lori, Alessandro. (2020). Are Self-Driving Cars Safe? Retrieved from https://www.verizonconnect.com/resources/article/are-self-driving-cars-safe/

Allssa, Walker. (2020). Are self-driving cars safe for our cities? Retrieved from https://www.curbed.com/2016/9/21/12991696/driverless-cars-safety-pros-cons

Autonomous Vehicles. Retrieved from https://www.ghsa.org/issues/autonomous-vehicles

Kalra, N., Paddock, Susan M. (2016). Driving to Safety- How many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability. Retrieved from https://www.rand.org/pubs/research_reports/RR1478.html

Allssa, Walker. (2018). It's time to delete Uber from our cities. Retrieved from https://www.curbed.com/transportation/2018/3/23/17153200/delete-uber-cities

Code output from Spyder Editor (results can vary from the Jupyter notebook)

1.**MSDS453_A2_Part1_8_TFIDF_CV_Doc2Vec**.py

DataFrame of the most important 6 terms.

A1. Analyst Judgement derived important terms using CountVectorizer.

Terms	A1-Freq
vehicle	11.273263
selfdriving	6.707892
autonomous	5.518257
company	5.103651
safety	4.186101
technology	4.121319

A2.TF-IDF derived important terms.

Terms	TFIDF-Freq
vehicle	0.238004
selfdriving	0.161158
autonomous	0.147179
company	0.140209
technology	0.104772
safety	0.104280

A2.Doc2Vec derived important terms.

A3-Doc2Vec-Count

Terms	
vehicle	9571
selfdriving	5695
autonomous	4685
company	4333
safety	3554
technology	3499

DataFrame of the lower importance 6 terms.

A1. Analyst Judgement derived least terms using CountVectorizer.

Terms	A1-Freq
engineer	0.620730
transit	0.619552
policy	0.619552
electric	0.613663
better	0.612485
rule	0.604240

A2.TF-IDF derived least important terms.

Terms	TFIDF-Freq
question	0.024738
rule	0.024702
government	0.024479
driven	0.023059
standard	0.022549
policy	0.020624

A2.Doc2Vec derived least important terms.

A3-Doc2Vec-Count

Terms

star 2

telemetry 2

canceling 2

techified 2

accordingly 2

innovating 2

Safety Count:334, Technology Count:515, Equal Count:0

Manually derived Class Terms and the derived Equivalent Classes from the data above.

Manually Derived Equivalent Terms

Class Terms

safety safety crash people driver human policy standard rule government testing

technology technology lidar autonomous research selfdriving electric engineer system

2. MSDS453_A2_Part9_10_ManualGeneratedLabels.py

Number of processor cores: 16

Count Vectorization. . .

TFIDF vectorization. . .

Doc2Vec Vectorization (50 dimensions). . .

Doc2Vec Vectorization (100 dimensions). . .

Doc2Vec Vectorization (200 dimensions). . .

TF-IDF/Random forest F1 classification performance in test set: 0.957

Count/Random forest F1 classification performance in test set: 0.969

Doc2Vec_50/Random forest F1 classification performance in test set: 0.956

Doc2Vec_100/Random forest F1 classification performance in test set: 0.956

Doc2Vec_200/Random forest F1 classification performance in test set: 0.931

F1 classification performance in test set with manual labeling

Algorithm	
CountVec/Random forest classification	0.969
TF-IDF/Random forest classification	0.957
Doc2Vec_50/Random forest classification	0.956
Doc2Vec_100/Random forest classification	0.956
Doc2Vec_200/Random forest classification	0.931

3. **MSDS453_A2_Part9_10_KmeansToGenerateLabels**.py

Number of processor cores: 16

TF-IDF Vectorization

TF-IDF Vectorization. . .

TF-IDF Vectorization K Means vectorization. . .

(679, 195)

Count Vectorization

Count Vectorization K Means vectorization. . .

Working on Doc2Vec vectorization, dimension 50

Doc2Vec 50 Vectorization K Means vectorization. . .

(679, 195)

Working on Doc2Vec vectorization, dimension 100

Doc2Vec 100 Vectorization K Means vectorization. . .

(679, 195)

Working on Doc2Vec vectorization, dimension 200

Doc2Vec 200 Vectorization K Means vectorization. . .

(679, 195)

F1 classification performance in test set with k-means labeling

Algorithm	
CountVec/Random forest classification	1.000
Doc2Vec_50/Random forest classification	0.994
Doc2Vec_100/Random forest classification	0.994
Doc2Vec_200/Random forest classification	0.994
TF-IDF/Random forest classification	0.949