Document Classification Model: NBA News Articles

Jason Lee

Northwestern University, SPS
Natural Language Processing
2020SP MSDS 453-56

Abstract

Information is power in the sports betting industry. When team news hits the web, betting syndicates need to be able to react with speed before market prices adjust. As web scrapers scour the internet collecting NBA news articles, a document classification model is needed to filter between the relevant information and irrelevant information. The goal of this project is to determine the optimal vectorization and training process needed to build a high performing document classification model. A factorial experiment will be conducted with varying vector lengths and three key vectorization methodologies: 1) Analyst Judegment 2) TF-IDF 3) Doc2Vec Embedding.

Keywords: Natural Language Processing (NLP), Document Classification, NBA, Sports Betting, Python

1. Introduction

For professional sports bettors, determining what information is relevant as they are flooded with hundreds of news articles and "breaking news" alerts covering numerous leagues across all different types of sports is crucial for long term success. As focused web crawlers scrape the internet capturing up to the minute news about each team, there needs to be an automated process to filter out the irrelevant articles. This is exactly what a document classification model can do.

A document classification model will be able to save a professional sports bettor countless hours by eliminating the manual effort needed to read the various news articles on each team and decide if it is useful. Building off of the corpus created by the focused web crawlers in the previous project, this project will center around the steps needed to create a finely tuned document classification model (Lee, 2020).

The predictive algorithm will be able to determine if a news article collected by the Spiders is relevant for the sports bettor or not. This automated filtering process, coupled with the focused web crawlers, will allow any sports bettor the ability to digest the right information in a timely manner.

However, the process of building a document classification model is nuanced with success being dependent on the training corpus, vectorization methodology, vector length, algorithm selection, and number of predicted classes. A central component of this project will be to determine the optimal vector length and vectorization methodology to feed into the document classification model.

Document vectorization is the operation used to convert a document of words into a representative, or meaningful, vector of numbers. This is a required step to enable machine learning algorithms the capability to perform mathematical calculations when training classification models. The length of these document vectors can be adjusted with varying degrees of success to the model's outputs.

A factorial experiment will be conducted with three different methodologies to the vectorization of the NBA news article corpus as the first factor and the length of the vectors as the second factor.

The three document vectorization approaches are as follows:

1. Analyst Judgment
2. TF-IDF
3. Neural Network Embedding

The lengths of the document vectors will be 50, 150, and 300. Each vectorization approach with each vector length will be evaluated by fitting a random forest classifier and comparing the model's accuracy metrics, using the F1 score as the primary metric for optimization.

1.1. Analyst Judgment

The Analyst Judgment methodology is a more "hands on" approach to document vectorization; it relies on the domain knowledge of the analyst. The analyst will need to determine which words should be the focal point in the vectorization process with two key criteria needing to be met, importance to the topic and prevalence throughout the corpus.

1.2. TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. The TF-IDF vertorization methodology is an algorithmic approach that involves using a weighted frequency count for words in the documents. This equation is used to determine which words are important and prevalent throughout the entire corpus. Each word in the document will have their TF-IDF score computed and a subset of the highest valued words across the corpus will be used in a vector to represent each document.

1.3. Neural Network Embedding

The Neural Network Embedding methodology will utilize the deep learning Doc2Vec algorithm provided by the Gensim package in Python (Řehůřek and Sojka, 2010). This approach is an unsupervised approach where the neural network is trained in a similar fashion as the Word2Vec algorithm but with an additional input vector used as a proxy document identification number. By the end of the training process, each document identification vector weights will have updated in a way that becomes representative of the collection of words used in the given document (Le and Mikolov, 2014).

The management problem this project addresses is the high cost of time and resources needed to manually filter through countless articles searching for information that could provide an edge for a professional sports bettor. Another issue addressed here is the speed to executing on this information. The sports bettor will need to react to important information before the markets have time to adjust. Automating this entire process with a document classification model will increase the return on investment for a professional sports bettors.

This natural language processing (NLP) project will also be a precursor for another NLP project focused on topic modeling using multivariate methods with unsupervised clustering algorithms, as well as multidimensional scaling. The output matrices from the three document vectorization approaches will be repurposed for the topic modeling algorithms. A.I. Sports is the financial sponsor for this project and the classification models built herein will be implemented through their company to server their professional sports betting clientele (Lee et al., 2018).

The following are the primary goals of this project:

1. Create a calibrated document classification model.
2. Determine which vectorization methodology produces the optimal results for this particular classification problem.
3. Produce document vectors that will be used in future projects.
4. Generate a reproducible Python workflow to easily share with colleagues.

2. Literature Review

Document classification models are powerful natural language processing (NLP) algorithms that enable automated sorting and filtering systems to function independently of human intervention. Document classification models are designed to digest an entire document and return a predicted class (Chollet, 2018). There are several data pre-processing steps and transformations that need to take place before a model can be trained.

2.1. Term Frequency-Inverse Document Frequency

The Term Frequency-Inverse Document Frequency (TF-IDF) algorithm is a relatively simple but powerful approach to document vectorization and classification modeling.

The actual equation consists of two components multiplied together as shown in Equation 1, where t is the given term. The first component is the term's frequency in the document (TF) and the second component is the term's inverse document frequency (IDF).

$$TFIDF(t) = TF(t) * IDF(t) \tag{1}$$

Term Frequency (TF) is a normalized count metric for a given term. It is calculated by counting the number of times a given term is used in a document divided by the total number of terms in the document as shown in Equation 2, where t is the given term and N is the total number of terms in the document (Weiss et al., 2015).

$$TF(t) = \frac{Freq(t)}{N} \tag{2}$$

Inverse Document Frequency (IDF) is calculated by taking the logarithm of total number of documents divided by the number of documents with the given term in it as shown in Equation 3, where D is the total number of documents in the corpus and $D(t)$ is the number of documents that contain the given t term (Weiss et al., 2015).

$$IDF(t) = \log \frac{D}{D(t)} \tag{3}$$

$$TFIDF(t) = \frac{Freq(t)}{N} * \log \frac{D}{D(t)} \tag{4}$$

2.2. Neural Network Embedding

Conceptually, a neural network embedding algorithm places word, or term, vectors into a geometric space where the Euclidean distance between words, or terms, creates context and meaning (Chollet, 2018). Once the terms are placed into a geometric space, mathematical equations involving words can be solved. For example, Equation 5 can be solved correctly with a properly trained neural network embedding model.

$$Utah\ Jazz - Utah + Miami = Miami\ Heat \tag{5}$$

This may be a simple equation for an NBA fan to solve, but understanding the words' relationship in this way for computers was a groundbreaking advancement. When working with documents, a popular neural network embedding solution is the Doc2Vec algorithm.

To understand the intricacies of the Doc2Vec algorithm, there needs to be an understanding of the Word2Vec algorithm. The Word2Vec algorithm, created by Thomas Mikolov at Google, is a two-layer neural network that utilizes the bag of words and skip-gram architectures in parallel (Mikolov, 2013).

2.2.1. Continuous Bag of Words (CBOW)

A Continuous Bag of Words network uses the nearby words to predict the target word (Lane et al., 2019). Figure 1 displays that framework to show how this algorithm works (Mikolov et al., 2013). The target word is in the middle having the two previous words and two proceeding words as the inputs.

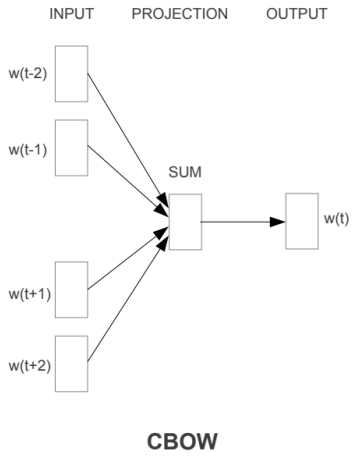

Figure 1: Framework of the Continuous Bag of Words model showing the inputs and output of a training sample.

This window of five words slides over one word to create the next training sample. Figure 2 provides a diagram showing how this works. The red word is the target word and the other four words are the inputs. The sliding window passes over the entire corpus creating a highly tuned model, especially with frequently used words (Lane et al., 2019).

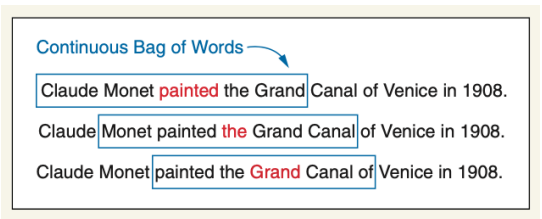

Figure 2: Sliding window of 5 words in the Continuous Bag of Words model.

2.2.2. Skip Gram

The Skip Gram algorithm is the reverse process as the continuous Bag of Words model (Mikolov et al., 2013). The input here is the given word and the goal is to predict the surrounding words as shown in Figure 3. The Skip Gram approach tends to perform better than the Continuous Bag of Words model when working with smaller corpora but takes longer to train (Lane et al., 2019).

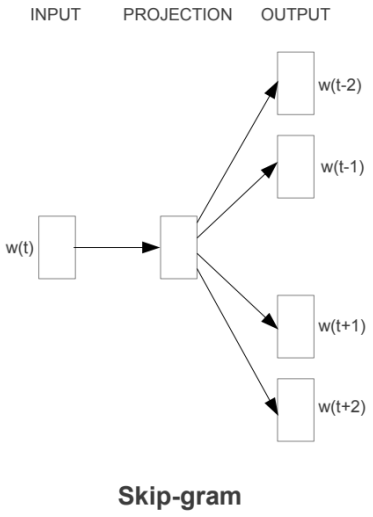

Figure 3: Framework of the Skip Gram model showing the input and outputs of a training sample.

2.2.3. Doc2Vec

The Doc2Vec algorithm behaves similar to the previous algorithms, except it adds an additional input vector representing the document.

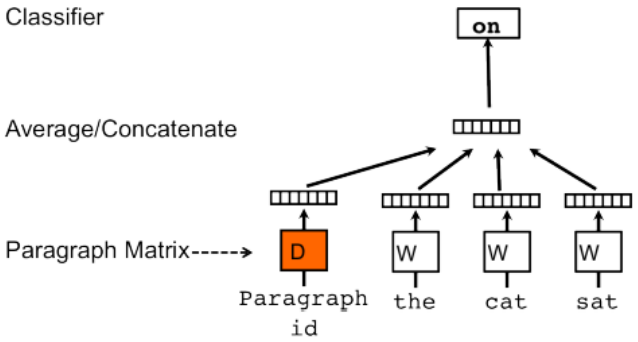

Figure 4: Doc2Vec training example

This document ID, or Paragraph ID in Figure 4, is fed into the model and the weights in the document ID are updated along side the word vectors from its document (Le and Mikolov, 2014). As the model converges during the training process, the document ID vector becomes Representative of the entire document in the new geometric

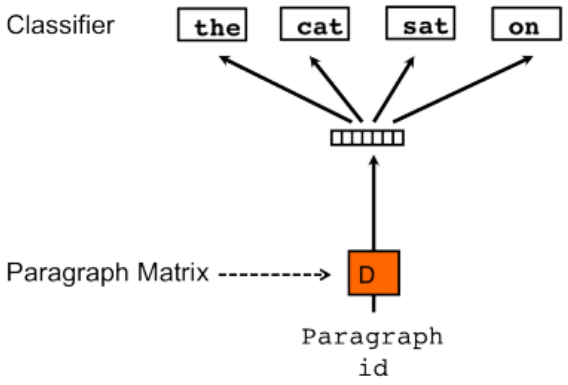

Figure 5: Doc2Vec prediction example

space. Each document now has meaning and documents closer together in the geometric space have more in common than documents farther away. Figure 5 shows how the document ID can be used in a similar fashion as the Skip Gram model.

The draw back is that the neural network embedding algorithms are computationally expensive to train compared to other vectorization methods (Lane et al., 2019).

3. Methodology

The methodology implemented during this project is as follows.

3.1. Document Corpus

The corpus of documents that will be used in this project was collected by focused web crawlers, or Spiders. The Spiders were released onto each National Basketball Association (NBA) team's official website moving from page to page collecting six important pieces of information from every news article they came across.

1. team = The name of the NBA team
2. url = The URL where the article is found
3. tags = The topic tags for the article
4. title = The title of the article
5. date = The date the article was posted
6. article = The complete news article

The topics contained in this corpus are wide ranging. There are articles written about the team's humanitarian efforts, potential trades, pre-game/match-up analysis, injury updates, post-game analysis, deep dive player specific topics, player written articles, front office management news, team perception/fan sentiment, fan outreach, and miscellaneous articles.

3.2. Dependent Variable

The document classification models built during this experiment will be designed to predict whether a news article provides relevant information for a sports bettor or not. The dependent variable is a binary flag.

$$1 = Relevant$$

$$0 = Irrelevant$$

Topics that are included in the positive class include potential trades, pre-game/match-up analysis, injury updates, post-game analysis, and team perception/fan sentiment. Everything else will be contained in the Irrelevant class.

3.3. Pre-Processing Corpus

The corpus will undergo extensive pre-processing before it will be able to be used in the experiment. The news articles are contain the raw text that was scraped from the websites creating

After the corpus was cleaned there were 214,655 tokens, with the unique count at 14,022 tokens. This will be the vocab size (V) in the experiment. Figure 6 shows the top ten most common words in the corpus.

Word	Frequency
game	3426
point	2141
season	1752
team	1621
get	1560
play	1534
nba	1528
say	1272
make	1225
time	1177

Figure 6: Top 10 words in the NBA news corpus

3.4. Factorial Experiment

In order to determine an optimal strategy for the vectorization of this corpus, a factorial experiment will be conducted. The experiment will consist of two factors with three levels each. The first factor will be the vectorization approach with the three levels as Analyst Judgment, TF-IDF, and Neural Network Embedding. The second factor will be the length of the vectors representing each document. The levels for the vector lengths are 50, 150, and 300.

There will be nine total document classification models built. The factorial experiment will hold all other parameters consistent allowing a clear interpretation of the interaction between the factors and the dependent variable. The nine classifiers will all be trained on the same random sample of documents, 70% of the total corpus. These classifiers will then be judged on their effectiveness predicting the classes of the 30% out of sample documents remaining.

4. Computational Experiment

The entire Python code for this project's factorial experiment will be attached to this paper, or can be reproduced by cloning the project's Google Colaboratory notebook at this url:

Google Colab Link

This script will begin by uploading and cleaning the NBA news article corpus followed by the experimentation with results. The experimentation portion of the Python code is an adapted version of the text classification example on the twenty newsgroup dataset (Miller, 2020). There is an additional section in the Python script providing various data visualisations of the corpus.

4.1. Analyst Judgment

The Analyst Judgment methodology to vectorizing documents involves using the word statistics from the corpus. To easily generate the word statistics for this project, the CountVectorizer function from the SKLearn package in Python was used (SKLearn, 2020a). This function tokenizes the terms in the document and proceeds to count the number of times each token was used.

Using the domain knowledge of the analyst, a judgment was made to include bi-grams in these counts because of the importance gained by including groupings of words that are common in a sporting context. For example, "shot" is one of the most common words in the corpus and alone doesn't provide much value. However, when using bi-grams, a "missed shot", "made shot", "bad shot", and "good shot" are frequent terms in the relevant class and "flu shot", "action shot", "hot shot", "trick shot", and "snap shot" are more likely to be found in the irrelevant documents. "Shot" on its own appears more often in the relevant class but there will not be as much discrimination between the classes as using a bi-gram with "shot".

Another example is the word "Free". There is not much discrimination between between the classes on its own but when a bi-gram is used there is informative data. A "Free Throw" occurs during a game, making it relevant, but a "Free Car" is a promotional ad that is irrelevant.

The CountVectorizer function was called three times to return a vector length of 50, 150, and 300. The returned vectors contain the number of times each term appeared in the document for the top 50, 150, and 300 terms respectively. Often this method produces sparse vectors but limiting the number of features will mitigate this issue (Brownlee, 2017).

The CountVectorizer only used the vocabulary in the training dataset to avoid any leakage or bias in the experiement. If there are terms in the test dataset that were not included in the training dataset then the function will simply disregard them.

4.2. TF-IDF

The Term Frequency-Inverse Document Frequency (TF-IDF) approach is similar to the analyst judgment in that they both use word statistics from the corpus. In fact, the CountVectorizer function followed by the TfidfTransformer will produce the same results as the TF-IDF equation.

For the experiment, the TfidfVectorizer function from the SKLearn package will be used to calculate the TF-IDF values for each term in the corpus (SKLearn, 2020b). Bigrams will be used in this approach to remain consistent with the Analyst Judgment approach.

4.3. Neural Network Embedding (Doc2Vec)

The final methodology to vectorizing the documents in the corpus is training a neural network embedding algorithm using Doc2Vec provided by way of the Gensim package in Python (Řehůřek and Sojka, 2010).

This methodology was produced substantially longer transformation and training times than the other approaches. The other approaches took roughly 0.7 of a second, while the Doc2Vec approach took 40+ seconds. That is 57 times longer.

5. Results

Each of the nine document classification models in this experiment were fit using a Random Forest algorithm. The out of sample test dataset with 231 documents was used as a measuring stick for each vectorization methodology and vector length. The F1 Score, harmonic mean between the precision and recall metrics, is the primary indicator used to evaluate the usefulness of each document classification model.

The F1 Score results for the factorial experiment are contained in Table 1.

Vector Length	Methodology		
	Analyst	TF-IDF	Embedding
50	0.835	0.838	0.821
150	0.844	0.822	0.767
300	0.820	0.818	0.743

Table 1: F1 Score - Harmonic Mean of Precision and Recall

The optimal document vectorization methodology for this particular document classification model is the Analyst Judgement. For every vector length, the Analyst Judgment approach reigned supreme. The Analyst Judgment with a vector length of 150 was the best performing classifier.

6. Discussion and Conclusions

The results from this experiment are not that surprising. The classification problem is a simple binary indicator in a domain where the relevant articles will all contain very similar words. The irrelevant class contains a wider variety of words and topics. The Analyst Judgment document vectorization methodology is able to correctly prioritize key words to attain strong discrimination between classes.

In conclusion, this project was able to accomplish each of the four goals originally laid out. A well calibrated document classification model was built that will be used by A.I. Sports when sporting events return from the COVID-19 hiatus. The first approach, Analyst Judgement, proved to be the most effective document vectorization methodology. The document vectors have been saved for future projects. And finally, a fully reproducible Python script was programmed that can be easily shared with others.

References

Brownlee, J., 2017. How to prepare text data for machine learning with scikit-learn. https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/. Accessed on 2020-05-01.

Chollet, F., 2018. Deep learning with Python. Manning Publications Co., Shelter Island, NY.

Lane, H., Howard, C., Hapke, H.M., 2019. Natural Language Processing In Action. Manning Publications Co., Shelter Island, NY.

Le, Q.V., Mikolov, T., 2014. Distributed representations of sentences and documents. ArXiv abs/1405.4053.

Lee, J., 2020. Focused web crawler: Nba team specific news articles. URL: https://github.com/papagorgio23/NBA_News_Spiders/blob/master/Focus%20Web%20Crawler%20Project.pdf.

Lee, J., Shephard, I., Wolande, P., 2018. A.I. Sports. https://aisportsfirm.com/.

Mikolov, T., 2013. Word2vec. https://code.google.com/archive/p/word2vec/. Accessed on 2020-05-01.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Miller, T., 2020. Twenty newsgroups vectorization and text classification. https://canvas.northwestern.edu/courses/112789/pages/twenty-newsgroups-vectorization-and-text-classification?module_item_id=1425573. Accessed on 2020-05-01.

Řehůřek, R., Sojka, P., 2010. Software Framework for Topic Modelling with Large Corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta. pp. 45–50. http://is.muni.cz/publication/884893/en.

SKLearn, 2020a. Countvectorizer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. Accessed on 2020-05-01.

SKLearn, 2020b. Tfidfvectorizer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed on 2020-05-01.

Weiss, S.M., Indurkhya, N., Zhang, T., 2015. Fundamentals of predictive text mining. 2nd edition. Springer, New York.