Recent Advances in the Application of Natural Language Processing to Unstructured and Semi-Structured Data in the Pharmaceutical Industry

Peter V. Henstock
Pfizer Inc., 1 Burtt Rd. Andover MA 01810

Abstract
Search and filtering methods are key technologies for unstructured or semi-structured texts. This paper focuses on a method to leverage lexical, morphological, semantic, and syntactic levels of linguistics to classify and rank text based on a semantic concept which is broader than a typical online query. Our approach leverages word2vec to identify the closest words followed by a random forest classifier. Although our approach can be applied broadly to many domains, the problem addressed in this paper is to rank order sentences in large corpus of online course transcriptions based on their need for human review for oversight.

Key Words: Natural language processing, word2vec, search, information retrieval

1. Introduction

The field of information retrieval can be tracked back to the 1950s as a requirement for identifying information in databases. With unstructured texts, the challenge is to enable queries to identify information without the formal schema of a database. As data volumes continue to scale now in the zettabytes [1], the problem persists since unstructured data may comprise over 95% of all captured data [2]. Natural language process methods are considered the key technologies to improving search capabilities.
This paper examines several different techniques that leverage different layers of the natural language processing pyramid (Figure 1). The approach, applicable to any domain or field of research, aims to identify text that matches a general concept area. The notion of a concept area differs from an everyday web search in that it contains a larger number of related words related to a semantic concept and aims to match against any text related to those words. The problem space is to prioritize portions of online course transcripts that should be examined as part of an oversight for the sales force.

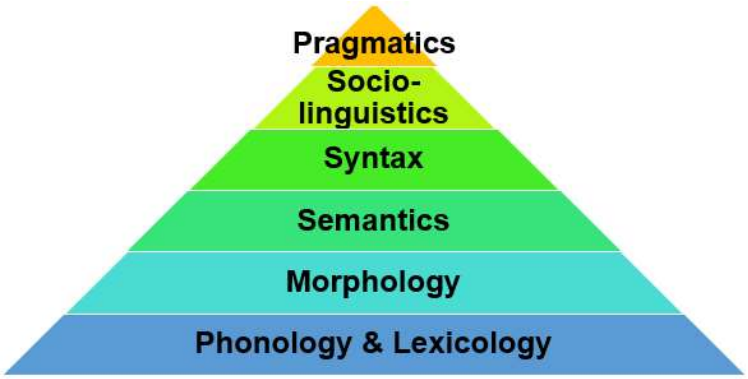

Figure 1: Natural language processing pyramid showing the hierarchy of linguistics.

Online education and training has increased dramatically over the past decade and is expected to reach $350 billion by 2025 according to Forbes post [3]. The number of online platforms for learning are increasing but include new standards and accreditation with sites such as Coursera, Udacity, Udemy, Lynda, Skillsoft and many others. When online courses are offered as part of a corporate curriculum, oversight becomes a challenge proportional to the number and scale of the courses offered. This is particularly true for the sales representatives that work in a highly regulated area as evinced by a lengthy list of marketing regulations [4] yet need to be trained with the science regarding the latest information on medicines [5], and be familiar with the regulations.

The problem statement is to score sentences within the course transcripts that potentially require human oversight. The machine learning problem is thus to develop a classifier that distinguishes innocuous sentences from those requiring human review. The classifier uses a list of provided words and phrases that represent the concept of bribery, quid pro quo exchanges and other inappropriate sales practices. The list contains 43 single words and 52 multi-word phrases. Many of the words in the list included wildcards to represent the different tenses or forms of a given word such as “brib*”. We refer to this list of terms in this concept group as the BadWordList.

2. Dataset

Since the goal of the project was to assess whether the algorithm could identify key phrases from transcripts, we extracted a set of course transcripts. We were able to download the transcripts from the Skillsoft site [6]. They offer a range of shorter courses online on different topics from business and marketing to computer science. We downloaded 639 courses for the dataset, extracted the content from the XML, and divided the data into a total of 373,008 sentences. Since we assume that most courses would be appropriate without issues, we generated this set to be the negative cases. For positive test cases, we requested a list of sentences that should be flagged as inappropriate. These were generated without specifically using the vocabulary list and totaled 42 sentences yielding a highly imbalanced dataset.

3. Lexical Experimental Approach

Using a standard lexical information retrieval approach of the vector space model [7], we generated a frequency table of hits against the BadWordList. The table had each sentence from the positive and negative cases as rows and the number of “hits” of occurrence of each term of the BadWordList which included wildcards and phrases. No stemming or lemmatization was used for the baseline result. The results are shown in Figure 2 using a receiver operating characteristic (ROC) curve to assess the classification at all thresholds. ROC curves are effective when handling imbalanced data sets since the results are normalized against the number of positive and negative examples. The single number summary of resulting curve is the area under the ROC curve (AUC) statistic [8].

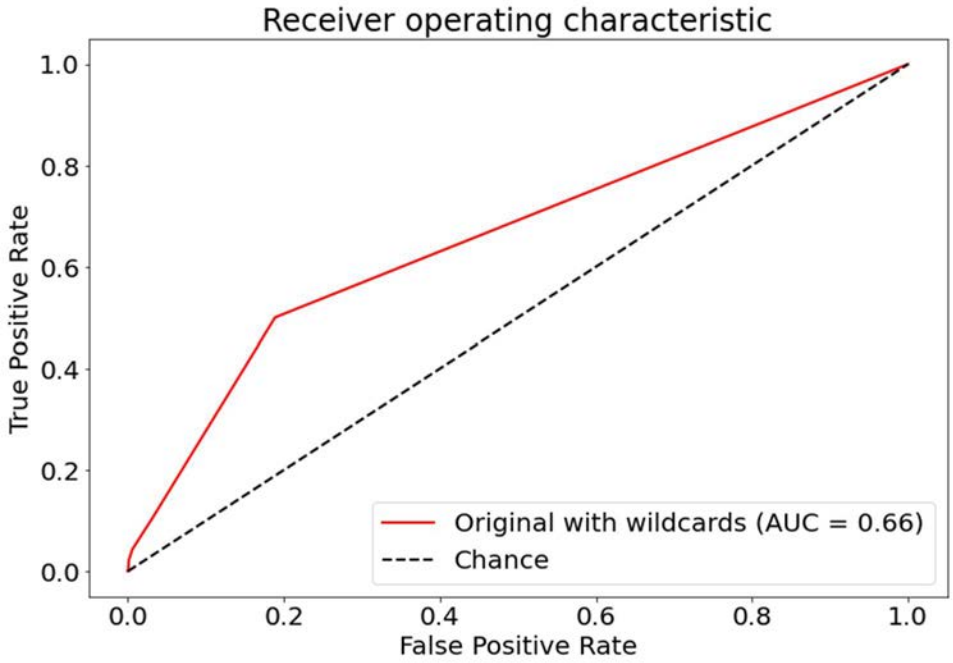

Figure 2: The original ROC shown in red relative the random 0.5 AUC dashed line shows moderate initial performance just matching the word list.

The fact that the BadWordList contained both individual words, words with wildcards (e.g. 'brib*') as well as phrases poses somewhat of a challenge for standard information retrieval engines. To make it work, we used regular expressions to match the various phrases which restricts the use of standard libraries. We explored whether the phrases, the non-phrase words, or the individual words from the phrase were the main drivers of the success. As expected, the combination achieves the best performance as shown in Figure 3.

The results at this point represent a scoring system where each words or phrases is considered as equal. That is, the match against a document sentence treats all words and phrases equally in scoring the match. However, all words certainly are not equal [9] when querying and many different weighting schemes have been developed dating back several decades [10]. The most common weighting scheme is TF-IDF or term frequency-inverse document frequency that is used in 83% of text-based recommender systems according to Beel et al.'s 2015 study [11]. The results from this approach show a modest improvement from a 66% to 68% AUC (Figure 4).

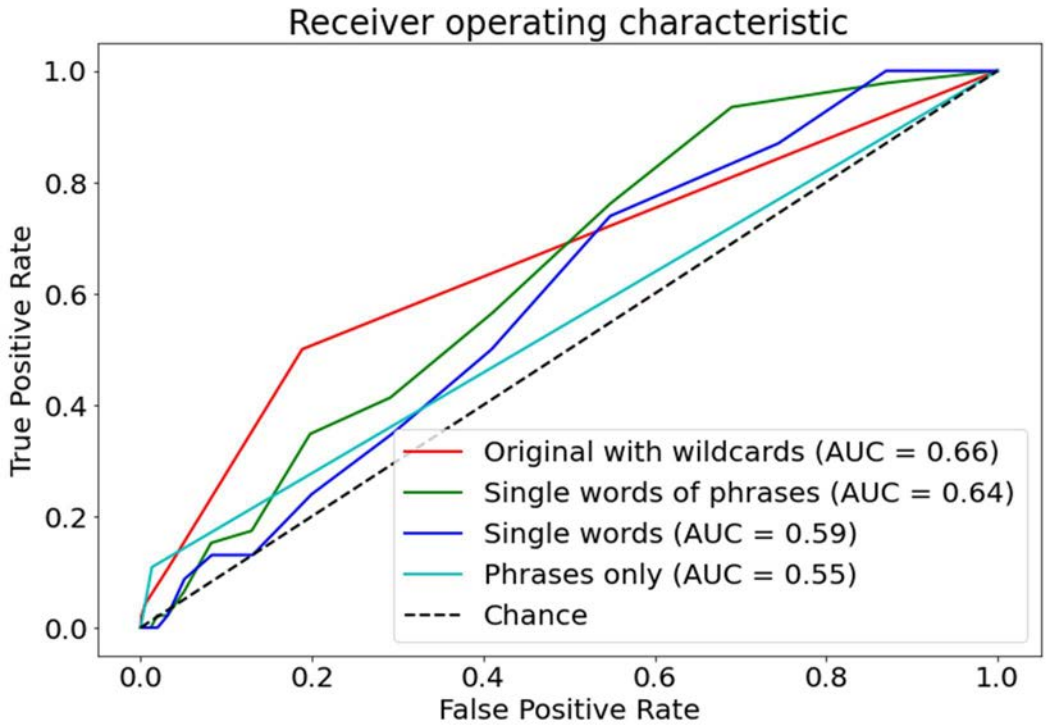

Figure 3: Results from Figure 2 are compared against single words extracted from the phrase words, the non-phrase single words, and the full phrases as separate queries.

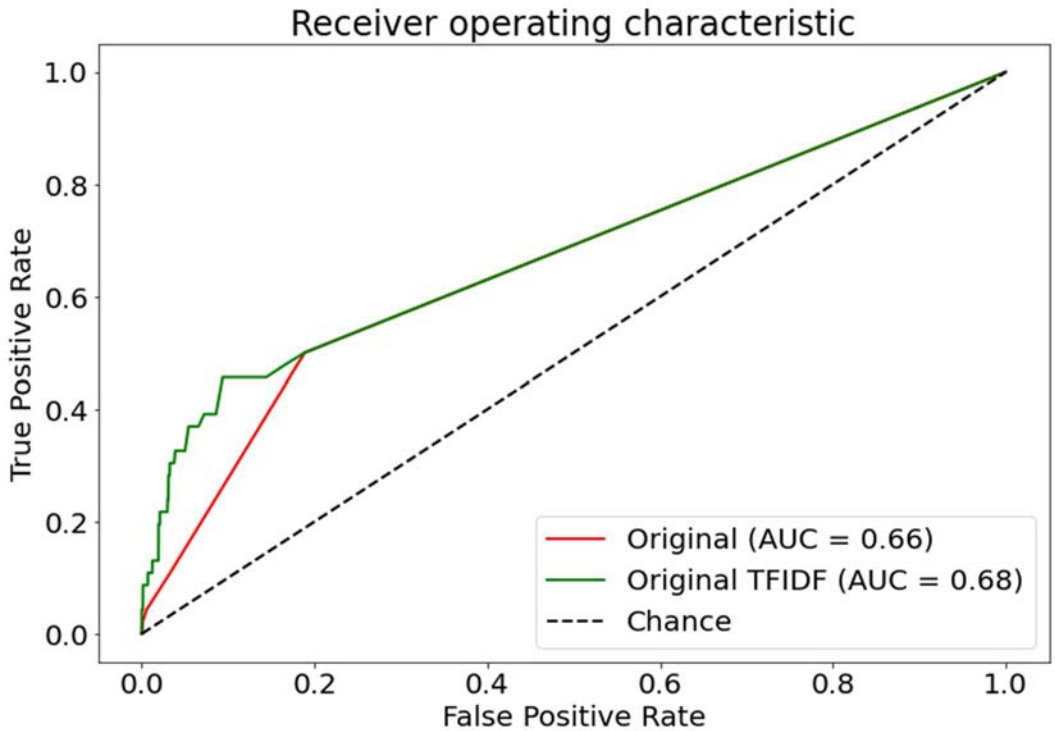

Figure 4: The use of term frequency-inverse document frequency (TF-IDF) improved results over the original.

4. Morphological Experimental Approach

Moving from the lexical level of the linguistics pyramid to the morphological level, the goal was to improve the matching between different versions of the same word. For instance, the terms bribe, bribes, bribed, and bribing refer to the same concept yet have different morphology in terms of their conjugation. Two strategies for dealing with these are stemming and lemmatization. Stemming refers to the removal of suffixes to produce a standard from of "brib" that recognizes the common endings and removes the variational endings, often producing a non-existent but standard form. Lemmatization has the same goal but aims to produce a proper word such as "bribe". Both stemming and lemmatization can be applied to nouns and verbs and use a range of different algorithms. They have been studied extensively such as in a comparison by Balakrishnan and Ethel [12].

The initial results of stemming and lemmatization using the Porter stemmer [13] and spaCy lemmatization [14] made little difference to the performance as shown in Figure 5.

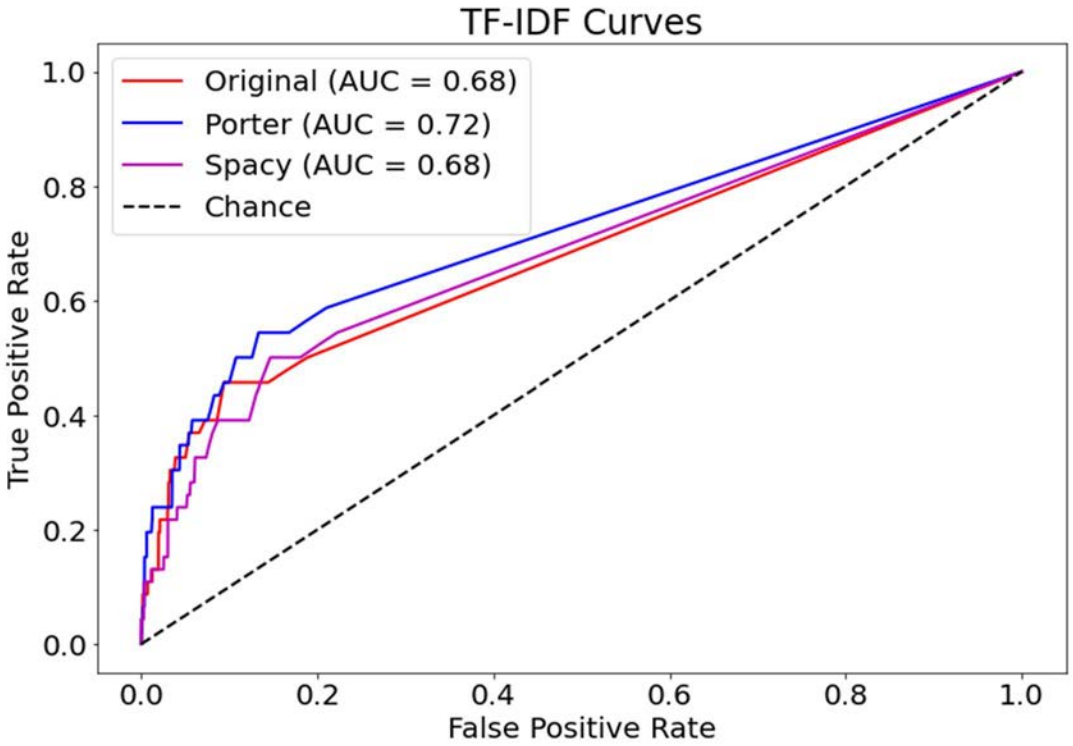

Figure 5: Porter stemmer and spaCy lemmatization versus the original version.

The morphological changes did not have the desired effect on the matching. Upon examining the resulting word matches, we determined that the forms of the original word list were not being properly modified. This was perhaps due to the lack of context. The simple solution was to manually add various forms of the original words into the matching list before passing them to the stemming and lemmatization. In addition to the Porter and spaCy approaches, the Lancaster and WordNet stemmers were evaluated. The Lancaster stemmer (also known as the Paice/Husk method) is a rule-based algorithm and known to be fairly aggressive in its stemming [15]. WordNet [16] is well established hierarchy of word definitions and word uses that includes various language tools, some of

which are captured in the Natural Language Toolkit (NLTK) [17]. The results improved substantially by adding the extra word forms into the search list (Figure 6). While the results were similar between all the methods, we selected the spaCy result going forward since it not only had the best result, but it is now considered the standard for industrial text processing.

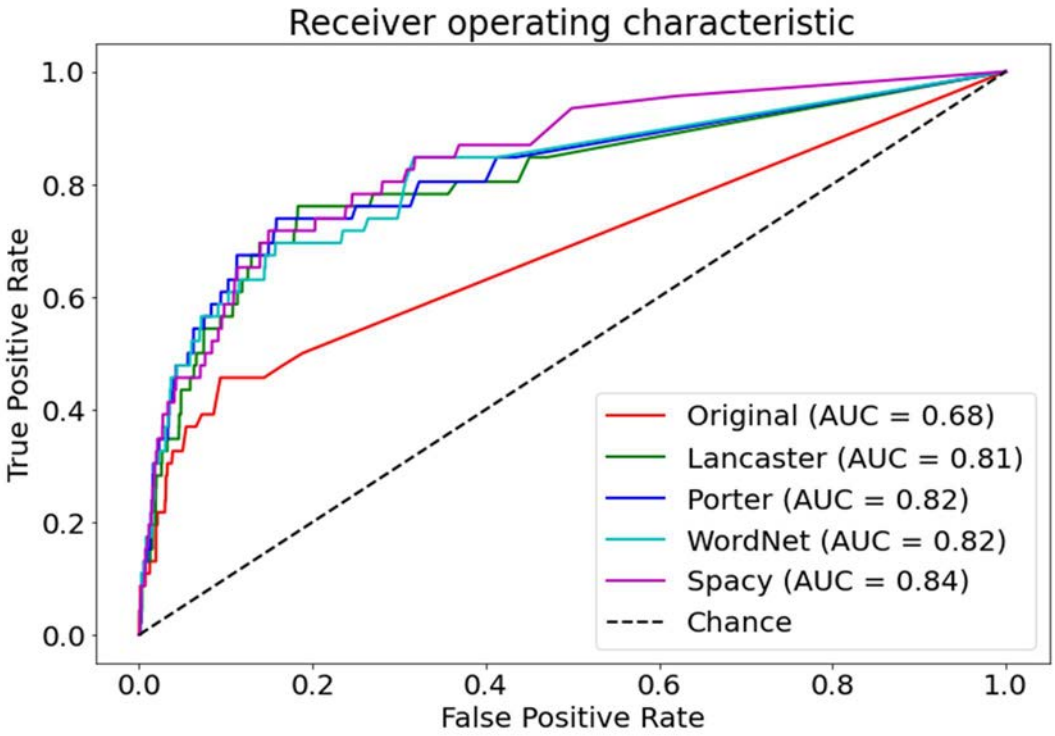

Figure 6: Results of stemming and lemmatization improve the results considerably.

4. Semantic Experimental Approach

With no further improvement likely using morphological approaches, the next aspect was to improve the word matching at the semantic level corresponding to the meaning of words. The challenge that the system faced was the limitations of the original word list. While the authors could work iteratively with any single data set and improve the result by adding words, the goal was to automate the process to achieve the best result without manual intervention. The first approach was to use the WordNet synonyms. Although we used WordNet for lemmatization, the primary value it has been offering the computational linguistics community is a lexical database with detailed lists of synonyms and antonyms, each detailed with parts of speech, use cases and definitions arranged in a hierarchy. We generated an automatic word list of all synonyms and antonyms for each entry in the wordlist. The results shown in Figure 7 show that no combination of these words improved the results. We believe that the addition of synonyms and antonyms included too many unrelated words and thus diluted the word matching. From an isolated list of words without context, the specific meaning could not be automatically established so the synonyms of all word senses were included. A semi-manual approach to select the appropriate synonyms may have yielded better results.

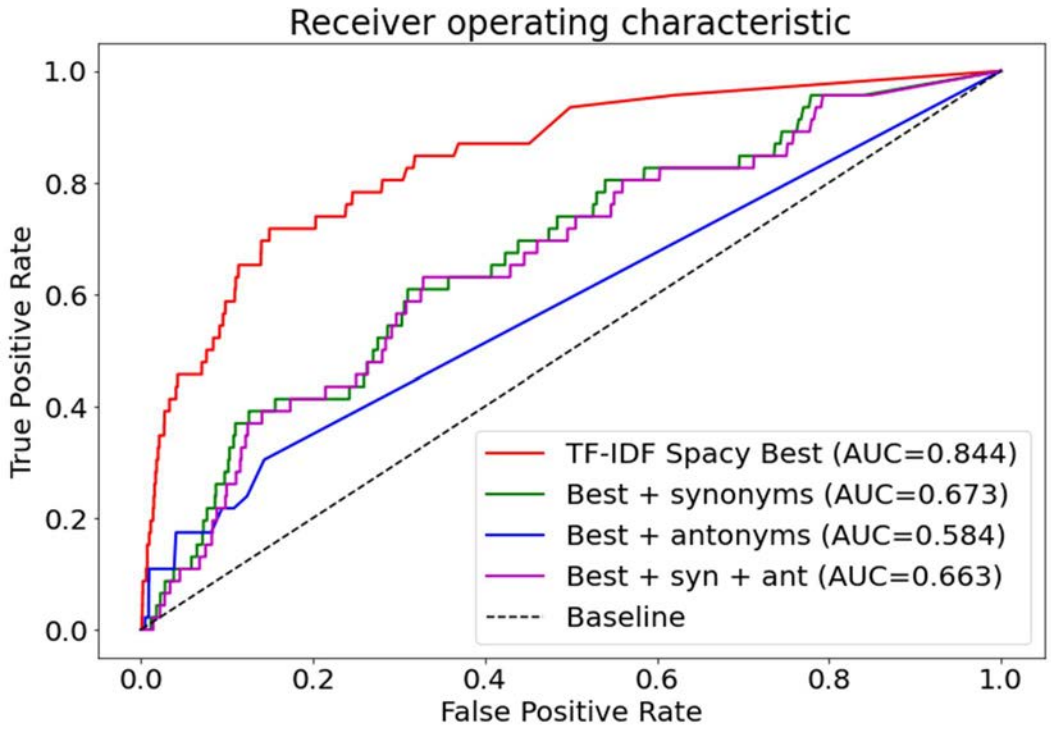

Figure 7: TF-IDF Spacy result compared against the addition of synonyms, antonyms, or both for each word in the wordlist.

4. Semantic & Syntactic Experimental Approach

The limitations of adding synonyms and antonyms directly led us to use the more recent and common approach of word2vec. This algorithm is a shallow neural network trained on a large corpus to predict words from context. The paper by Mikolov et al. [18] uses the Common Bag of Words (CBOW) or Skip-gram algorithm to create a vector for each word by training it on the surrounding words within a specified window shown in Figure 8. The CBOW uses the surrounding words to predict the word in the middle of the window. The Skip-gram approach does the opposite of predicting surrounding words from a given word and is generally considered more efficient. We used a pre-trained word2vec version trained on 3 billion words from Google News and a 300-dimensional word vector available from Google's code archive [19]. With this approach, words have been trained to predict their context but the value is in the middle box of each of the algorithms that represents the context of each word that represents the syntactic and semantic function of the word. Captured in a 300-dimensional vector, the function can be compared to other words using distance measures.

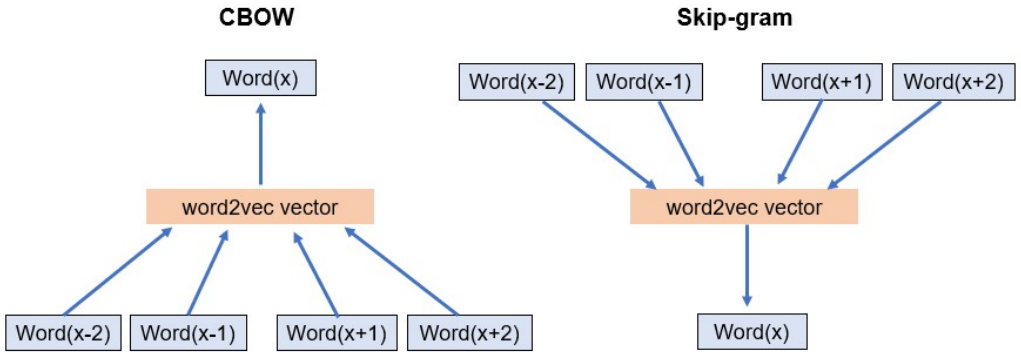

Figure 8: The word2vec diagram of Continuous Bag of Words (CBOW) and Skip-gram.

The distances between words are useful for comparing individual words, but our goal is to rank the sentences. We can look up each of the words in our wordlist and we can look up each word for any given sentence which has a variable number of words. We now need to compare two sets of 300-dimensional vectors of different lengths. A common approach is to use the Word Mover distance (WMD) [20] which is a language-based version of the perhaps better known earth mover's distance (EMD) [21]. The WMD compares the set of vectors for a sentence against the set of vectors from the wordlist and thus scores each sentence based on its similarity as illustrated in Figure 9.

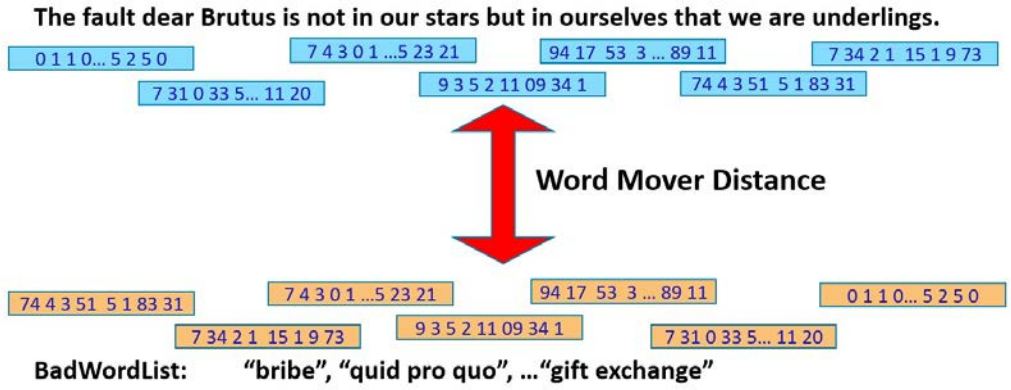

Figure 9: Word mover distance approach to score the vectors of a sentence against the vectors of the BadWordList.

The results of the Word Mover distance are shown in Figure 10 are rather disappointing. Like the addition of synonyms using WordNet, the addition of vector matching appears to reduce the overall performance. We believe that the variety of words that occur in the BadWordList tend to match against many words that are not related. We are not looking for the best match to every word but merely to a subset of the BadWordList so the full Word Mover distance may be the wrong measure.

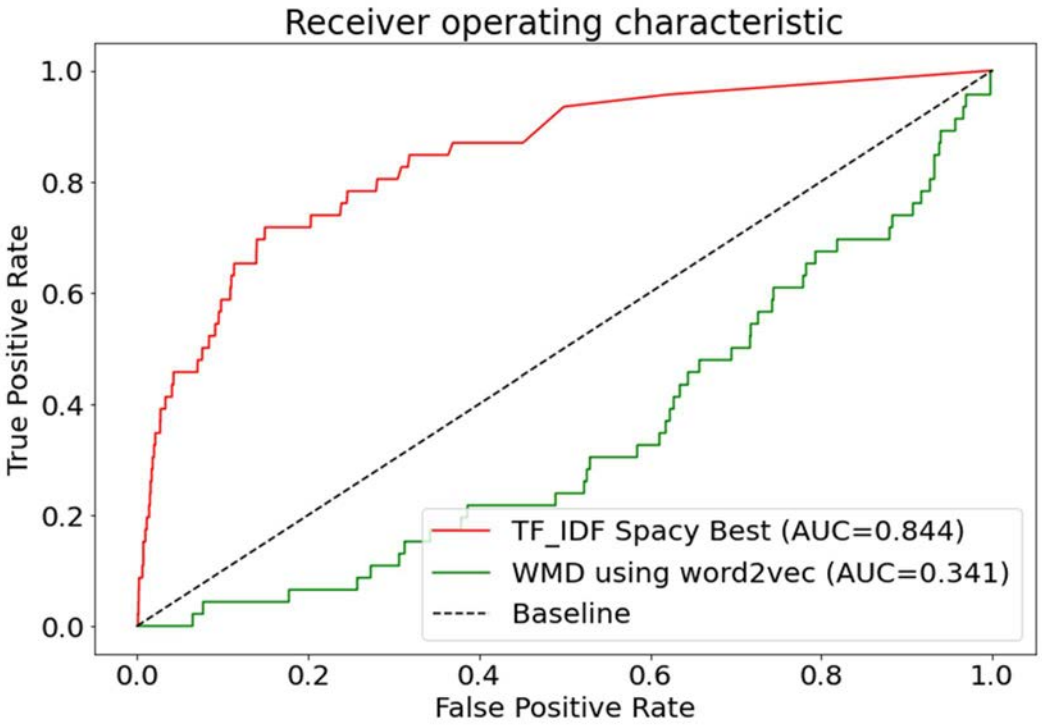

Figure 10: Word Mover distance results using word2vec against the best model.

To resolve the issues with the general matching of word2vec, we introduced a ranked matching of vectors. Rather than match the entire set of vectors against the full set of vectors associated with the BadWordList, we computed the top k closest vector distances as shown in Figure 11. The score for a given sentence is the kth closest distance between pairs of word2vec vectors. Sentences that discuss "gift" but not bribe can be matched based on that subset of words in the BadWordList rather than the match against the full set of words. The results in Figure 12 do improve over the word mover distance, but are still not as good.

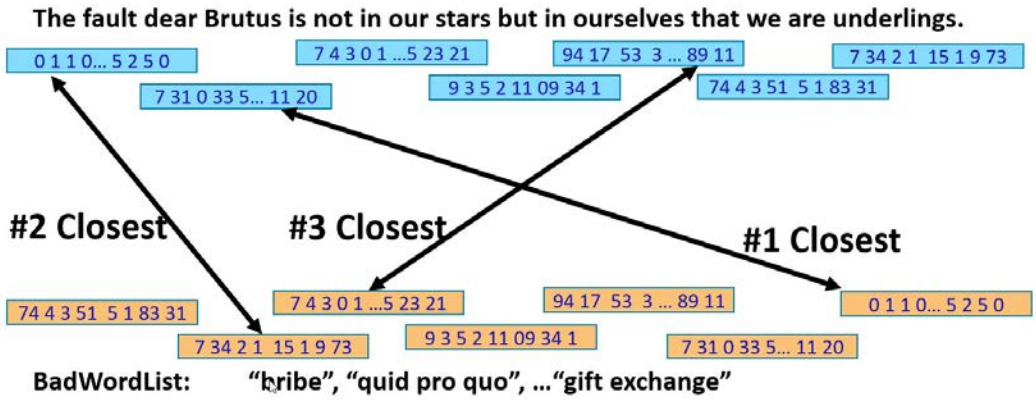

Figure 11: Matching closest word2vec vectors between sentences and the BadWordList.

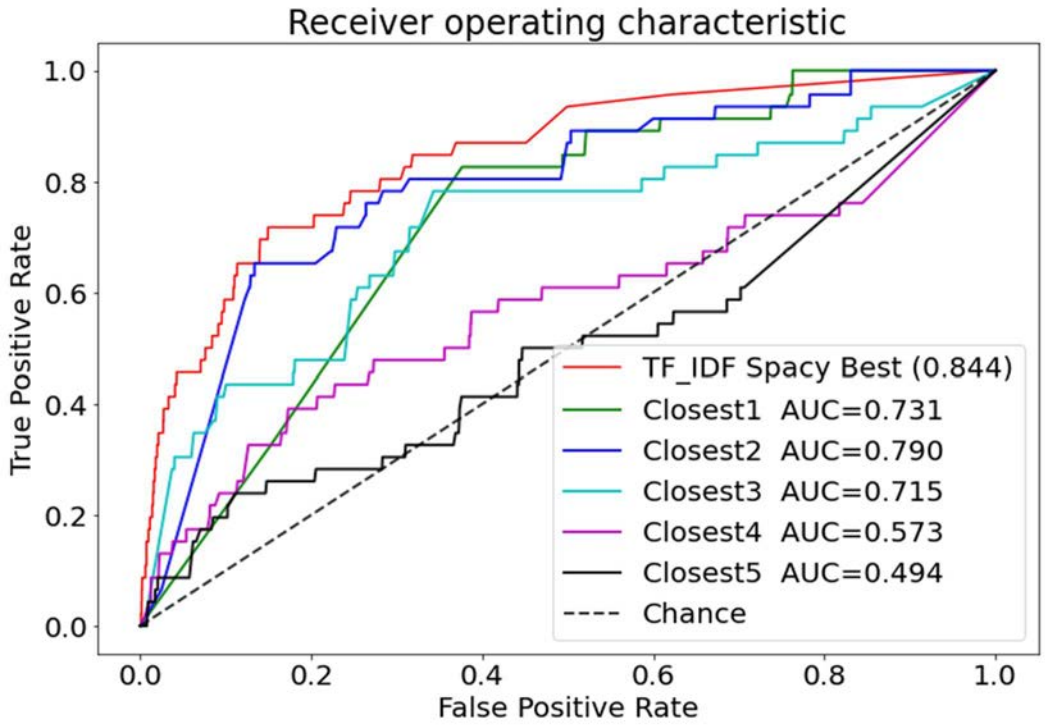

Figure 12: Matching the closest k word2vec vectors between a sentence and the BadWordList terms.

The third variation of the word2vec distances is shown in Figure 13. The previous version ranked the matches of all pairs of words. In this case, a single word from the BadWordList could match multiple words in a sentence and achieve a high score. This new variation identifies the closest match of each word in the BadWordList against all words in the sentence. In the previous cases, a close matching word could dominate the score whether that was appropriate or not. In this method, every word in the BadWordList contains a score which balances the weighting. Many of the words will achieve a neutral score but are still included but the inclusion of more words balances the results. In addition, each sentence, regardless of its length, will produce a vector of distances—one entry for each word in the BadWordList. Instead of using the distances directly, the distances are used as a feature vector for a training and testing approach using the random forest classifier [22] with default settings from the Python scikit-learn package [23].

The results for this approach are shown in Figure 14. Unlike previous versions, this approach was evaluated using a 5-fold cross-validation and achieves an average AUC of 0.96 for this problem which is considerably higher than the previous best result with an AUC of 0.84. Unlike the previous iterations, the use of the training-testing paradigm facilitates the incremental improvement of the model as additional training labels are applied through user feedback.

Figure 13: The scores are computed as the average of the closest matches for each word in BadWordList against the sentence.

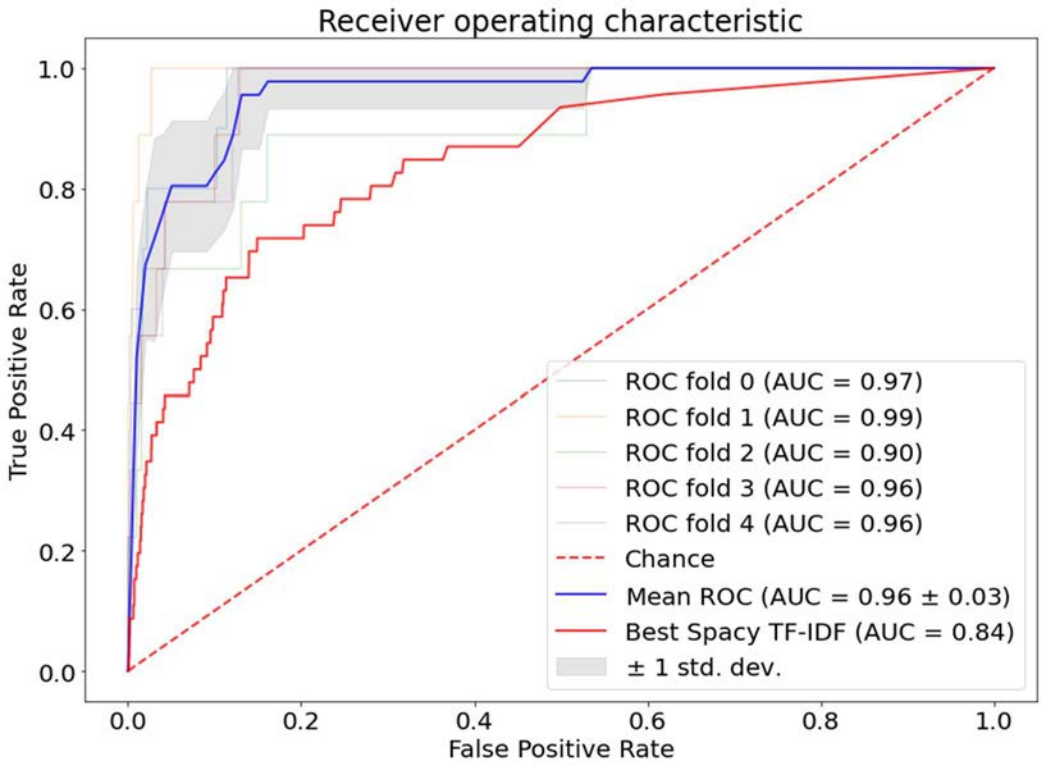

Figure 14: Random forest classifier trained on feature vector of closest distances to each word in the BadWordList. The results use a 5-fold cross-validation.

4. Conclusion

The ability to find relevant texts that match specific criteria is a common information retrieval problem. The traditional methods that utilize lexical counting of frequency-weighting of words provide a standard but have much room for improvement. This problem addressed in this paper addresses a ranking of course issues by classifying

sentences that are potentially problematic from an oversight perspective. The results with an AUC of 0.96 can effectively filter and prioritize the text needed for human review which is important given the volume of online training content. Although the problem was specific, the solution is equally applicable to other problems and fields where a curated set of search terms define a semantic concept as the criteria for matching. The use of morphological stemming and lemmatization and expansion of the list of words defining the concept proved a critical factor. The other main boost in performance was achieved by converting the sentences into a feature space using the closest word2vec distance for each of the search terms. The word2vec added both a semantic and syntactic context for the queries and converted the problem into a machine learning training-testing paradigm. With this paradigm, it is trivial to continuously improve the results by adding additional training data through feedback.

References

1. *New Digital Universe Study Reveals Big Data Gap Less Than 1 of World's Data is Analyzed Less than 20 is Protected*. 2012, Dell Technologies.
2. Gandomi, A. and M. Haider, *Beyond the hype: Big data concepts, methods, and analytics.* International Journal of Information Management, 2015. **35**(2): p. 137-144.
3. Koksal, I., *The Rise of Online Learning*. 2020, Forbes: Forbes.
4. *Marketing and Advertising of Pharmaceuticals*. 2018; Available from: https://www.ncsl.org/research/health/marketing-and-advertising-of-pharmaceuticals.aspx.
5. Bauer, R.A. and L.H. Wortzel, *Doctor's Choice: The Physician and His Sources of Information about Drugs.* Journal of Marketing Research, 1966. **3**(1): p. 40-48.
6. *Skillsoft*. 2020, Skillsoft: https://www.skillsoft.com.
7. Zhai, C. and S. Massung, *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. 2016: Association for Computing Machinery and Morgan & Claypool Publishers. 471.
8. Flach, P. and J. Hernandez-Orallo. *A Coherent Interpretation of AUC as a Measure of Aggregated Classification Performance*. in *28th International Conference on Machine Learning*. 2011. Bellevue, WA: ACM.
9. Rajaraman, A. and J.D. Ullman, *Data Mining.* 2011: p. 1-17.
10. Robertson, S.E. and K.S. Jones, *Relevance weighting of search terms.* Journal of the American Society for Information Science, 1976. **27**(3): p. 129-146.
11. Beel, J., et al., *Research-paper recommender systems: a literature survey.* International Journal on Digital Libraries, 2015. **17**(4): p. 305-338.
12. Balakrishnan, V. and L.-Y. Ethel, *Stemming and Lemmatization: A Comparison of Retrieval Performances.* Lecture Notes on Software Engineering, 2014. **2**(3): p. 262-267.
13. Porter, M.F., *An algorithm for suffix stripping.* Program, 1980. **14**(3): p. 130-137.

14. Honnibal, M. and I. Montani, *SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.* To appear, 2017.
15. Paice, C.D., *Another stemmer.* ACM SIGIR Forum, 1990. **24**(3): p. 56-61.
16. Miller, G.A., *WordNet: a lexial database for English.* Communications of the ACM, 1995. **38**(11).
17. Loper, E. and S. Bird. *NLTK: The Natural Language Toolkit*. in *ACL Workshop on Effective Tools and methodologies for Teaching Natural Language Processing and Computational Linguistics*. 2002. Philadelphia: ACM.
18. Mikolov, T., et al. *Efficient estimation of word representations in vector space*. 2013. 1-12.
19. *Word2vec*. Google Code Archive 2013; Available from: https://code.google.com/archive/p/word2vec/.
20. Kusner, M.J., et al. *From word embeddings to document distances*. in *International Conference on Machine Learning*. 2015. Association of Computing Machinery.
21. Peleg, S., M. Werman, and H. Rom, *A unified approach to the change of resolution: space and gray-level.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989. **11**(7): p. 739-742.
22. Breiman, L., Machine Learning, 2001. **45**(1): p. 5-32.
23. Pedregosa, F., et al., *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 2011. **12**(October): p. 2825-2830.