# Unraveling the Elements of Effective Altruistic Appeals through Machine Learning and Natural Language Processing

Sourav Yadav[1], Sankalp Arora[2], Akash Kumar[3], and Kaveri Verma[4]

souravyadav988@gmail.com[1], sankalparora5@gmail.com[2], akash0697@gmail.com[3], vkaveri10@gmail.com[4]

**Abstract.** In today's world, online platforms such as social media, philanthropic communities, and Q&A websites provide opportunities for people to be altruistic by donating money or answering questions without expecting anything in return. The r/Random_Acts_Of_Pizza subreddit on Reddit is one such online community where users can post requests for free pizza while explaining their current situation, and the outcome of each request is either successful or unsuccessful. This study seeks to explore the determinants that impact the outcome of such selfless appeals. To achieve this, we propose a new model architecture that combines two models, one that deals with sparse text vectors and the other that analyzes dense features from previous works to predict the success of a request. The study reveals that the probability estimated from the first model and the number of comments on the request post is crucial in predicting the outcome of a request.

**Keywords:** Natural Language Processing, Machine Learning, Feature Engineering

## 1 Introduction

The amount of information shared in the form of text ever since the dawn of the internet has been appalling. In contemporary times, written information is a pivotal, if not the paramount, reservoir of knowledge. Efforts to get information out of text (or speech) have been underway since the 1950s when in the Georgetown-IBM Experiment, scientists successfully translated 60 Russian sentences into English and made some progress toward machine translation [1]. Until the 1980s, the research work under Natural Language Processing was based on sets of hand-written rules, which became obsolete after the introduction of machine learning algorithms to textual data. The beginning of statistical NLP gave way to the advancement of many different subfields like machine translation, text classification, speech recognition, etc. Further, demand for advanced maneuvers in many natural language processing applications gave rise to what is today known as Neural Natural Language Processing or Neural NLP [2]. While utilizing deep learning methods, Neural NLP has become increasingly essential for sectors like healthcare and medicine.

Recently, a wave of use cases in Computational Social Science has surfaced, requiring state-of-the-art NLP tools to solve them [3]. Be it determining biased news, identifying emotion behind tweets, or assessing the persuasiveness of an argument in a debate, the use of NLP tools for social causes has clearly increased. With the popularity of social media and Natural Language Processing in a social setting, the rise of philanthropic communities on the internet is notable. Websites like Kickstarter, Ketto, Reddit, and Kiva have made their presence immense and facilitated many people worldwide to get out of unpleasant situations via crowdfunding. For example, the Indian online crowdfunding platform, Ketto, has been responsible for raising over Rs. 2000 crores across 3.2 Lakh online fundraisers. This new era of online fundraising has undoubtedly helped people around the world and will certainly continue to do so. Our project focuses on a similar community forum, under the Reddit website, known as r/Random_Acts_Of_Pizza [4]. The forum was created in the year 2010 with the aim of getting pizza to those who need it. Requesters could explain their situation in a post, giving information about why they need a pizza and how it would help them. It would then depend on the other users to donate a pizza to the requester.

Our study relied on information mentioned on the CS Stanford website, and the foundation for this project has been the work done by Althoff et al. [5]. The dataset contains 5671 requests collected from the Reddit community between December 8, 2010, and September 29, 2013. There are 33 features present in the dataset, including a target variable containing information about whether the requester received the pizza or not. The project proposes an architecture of a two-stack binary classification model to predict the success of an altruistic request. The first model deals with the textual data by utilizing textual feature extraction techniques to convert the raw text data into vector form and then classify the vector data against the target variable. The finished model is then used to estimate the probabilities of the request being successful by the text. This probability estimate is then used as a dense feature among other features, like account_age of the requester, the number of likes and dislikes, etc., to predict request outcomes. We have then used the final model to determine the importance of each feature and how they contribute towards predicting the target variable.

## 2 Literature Review

In recent years, along with the increase in the research work under NLP in a social setting, there have been multiple attempts to decode linguistic factors in textual data and how they affect the success of requests. With the objective of interpreting what makes a favor successful, Althoff et al. [5] discuss high-level social factors, like the status of the requester and the similarity between the requester and the donor, textual factors, like evidence of need and the sentiment of text, and, finally, temporal factors like age of the account and the time the request was made. A logistic regression model then binds these factors together to classify whether the request was successful or not. The model also allows the researchers to reason out the significance of independent variables in predicting the dependent variable. Similar to the base paper, Jacek and Sabrina et al. [6] critically evaluate different algorithms over the RAOP dataset. In addition to the social and linguistic factors, such as the requester's identity and how the requester is asking, the researchers also aim to extend the work by investigating the sentimental factors satisfying a donor.

H.-P. Hsieh et al. [7] extend the base paper by introducing additional features like Topic, Role, and Centrality. By employing Bag of Words & N-Gram, along with considering the requester's interaction with other users, these characteristics effectively addressed the underlying subtext of the text. Furthermore, a groundbreaking model called the Graph-based Predictor for Request Success was utilized in this context. Utilizing a Request Graph and a Propagation-based Optimization algorithm, this approach captures the interrelatedness of features among requests within the dataset. By learning feature weights, it calculates the probabilities of requests being successful or unsuccessful. Specifically, requests with higher similarity scores are assigned the same label, further enhancing the accuracy of the model. Ahmed et al. [8] discuss the same method, focusing more on the Topic and Role as additional features.

Durmus and Cardie et al. 's [9] work is based on debate.org, which provides a platform for people to express their opinions and beliefs. It takes on the task of judging which debater will be more successful. This research paper introduces a novel dataset to explore the impact of linguistic usage versus pre-existing beliefs on persuasive communication. It also proposes a controlled framework that incorporates the political and religious ideologies of the readers, thereby providing a structured environment for analysis. It inculcates user-based features like ideology and opinion similarity and linguistic features like text length, sentiment, and features gathered using TF-IDF. Similarly, Yang et al. [10, 11] also revolved around constructing a framework for modeling persuasive tactics. This involved the utilization of a semi-supervised hierarchical neural network that incorporates attention mechanisms at both the word-level and sentence-level. The aim was to assess the degree of persuasiveness achieved quantitatively.

## 3 Proposed Architecture and Steps

This project utilizes the crucial features gathered from the pre-existing dataset as well as the previous works in the literature review while framing our architecture consisting of a stack of two models to deal with the textual and numerical data separately. In this section, we will be stating the architecture developed and the process followed in creating the two models.

### 3.1 Data Exploration

Data exploration allows for a better understanding of the dataset, making it easier for us to navigate the data better and select methods suitable for that

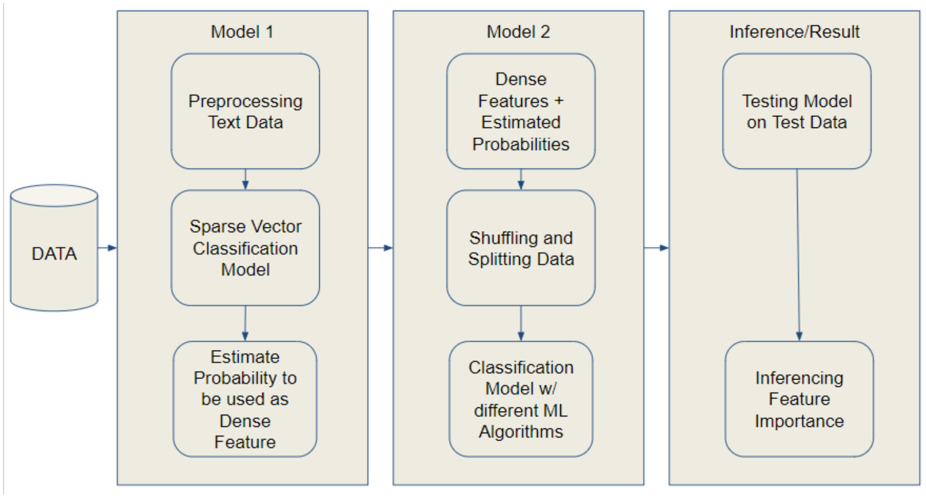
DATA
Model 1
Preprocessing Text Data
Sparse Vector Classification Model
Estimate Probability to be used as Dense Feature
Model 2
Dense Features + Estimated Probabilities
Shuffling and Splitting Data
Classification Model w/ different ML Algorithms
Inference/Result
Testing Model on Test Data
Inferencing Feature Importance

**Fig. 1.** Model Architecture

data. After a simple exploration of the RAOP dataset, we notice that there are 5671 requests with an average success rate of 24.63%, making it an imbalanced dataset. This will allow us to choose a Stratified K-Fold instead of the standard K-Fold Cross Validation technique while training our models. Stratified K-Fold Cross Validation partitions the dataset so that the validation set will have an equal number of instances of the dependent variable [12]. This ensures that no instance will overpower the other in the validation set.

### 3.2 Text Preprocessing

*Lowercasing, Removal of Punctuation & Stop-words* are a few of the most common preprocessing steps that allow us to reduce the redundancy between words. Similarly, Punctuation in a text can sometimes create difficulty in comprehending the text or extracting tokens. Therefore, it becomes necessary in some use cases to remove the punctuation in the text.
Stop-words are widely used irrespective of the language and are usually removed from the text as they lack constructiveness to the text. Therefore, our study employed NLTK [13] list of English stop-words in our project to remove them from the request.

*Regular Expression* [14] has proven powerful in searching and manipulating text strings and has been an essential tool in text analytics. We can use it to find patterns in the text that we want to remove or edit. For example, in the project, we used the "re" python library to find and expand the contractions such as "don't," & "can't," "do not," & "can not."

*Tokenization* is the fundamental task of the NLP preprocessing pipeline in which we break a text sample into tinier components. These units might be words, subwords, or characters. We have used the "regexp_tokenize" function from the NLTK [13] library to tokenize the cleaned text.

*Lemmatization* is the most crucial text preprocessing step that stems the word while ensuring it does not lose its meaning. The project uses the Wordnet Lemmatizer present inside the NLTK [13] package. Wordnet [15] is a large lexical database that aims to establish structured semantic relationships between words. We utilize the NLTK [13] interface of this database using the function lemmatize() of the instance WordNetLemmatizer().

## 3.3 Feature Engineernig

We split the numeric and textual data and separately worked on it for the two different models.

**Numeric Data** Based on the previous literature and knowledge, we manipulated the dataset in order to get the features. We use two features to understand how any posted request is doing in the community by subtracting the number of downvotes from the number of upvotes at retrieval as well as getting the number of comments at retrieval. We also add the length of the text as a feature to check if longer texts lead to the request being successful or not.
An "Evidence" feature is calculated by setting a value of 1 if there is a presence of any image link and 0 if there is no link present [5]. For other features, we consider the account age of the requester and whether the requester has posted before in the RAOP subreddit. These features give us some information about the requester.

**Text Data** We combine the title of the request and the main text of the request to get the final text which can then be converted into vectors using One-Hot Encoding [16], TF-IDF [17], Word2Vec [18], and Doc2Vec [19] techniques.

*One-Hot Encoding* or Count Vectorizing is the method of representing words by creating a vector having as many dimensions as the unique words in the vocabulary. Inside the vector, if the text data features that word, we put a "1" in that dimension; otherwise, we put "0". This gives us a huge sparse vector representation of all the text requests in the data. However, one of the disadvantages of the Count Vectorizing technique is its inability to represent the semantic and statistical relationship between the data.

*TF-IDF* vectors feature statistical representation of text regarding the word's importance or relevance in a document. This representation is constructive in information retrieval systems as it allows the system to look for more relevant words in the search query. TF-IDF [17] measure is the product of Term Frequency(TF) [20] and Inverse Document Frequency(IDF) [21], where TF looks at how many instances a word is repeated, while IDF indicates the relevance of the word. Therefore, it gives weightage to each word based on its frequency in the corpus.

*Word2Vec* frequently used embedding techniques which aim at learning embeddings for every word in the corpus[22]. Word2Vec helps in capturing the semantic representation of the text. It uses neural networks to derive the word's meaning from its context. Therefore, Word2Vec projects the meaning of the word in a high-dimensional vector space and clusters the word with similar meanings together. We used the pre-trained Word2Vec model over the Google News Corpus for this project [18].
In order to find the sentence vector by using individual word vector representations, we calculate the average of the vectors of words in the text. Another method that has given good results in the past is calculating the minimum and maximum from the list of vectors of words in the text and concatenating it to form a more extensive vector representation [23].

*Doc2Vec* [19] is very similar to the Word2Vec [18] model in terms of its general architecture. In addition to learning the word vectors, it also learns the paragraph vector associated with the full text. There are two architectures of the Doc2Vec model but this project uses the Distributed Memory (DM) architecture of the Doc2Vec Model packaged under the Gensim library.

### 3.4 Modelling

This section briefly describes the stack of two models present in our classification architecture. The architecture consists of two models, one that deals with the high-dimensional sparse vector converted from the textual data and the other model that deals with the dense features pre-existing in the dataset as well as the ones engineered by previous literature.

**Text Model** The request title and the request text are combined with the target variable into a separate data frame. This data frame will be helpful in converting each row of the text data into a vector using different text embedding techniques mentioned in section 3.3. The resulting vectors from the different embedding techniques are then fed into different machine learning algorithms like Logistic Regression [24], Gaussian Naive Bayes [25], and Random Forest [26]. (Note that we do not use K Nearest Neighbors [27] or Support Vector Machines [28] as they do not give a probability estimate needed for our second model.)
Based on the accuracy of the different models created by the combinations of different text embedding techniques and machine learning algorithms, we select the models to estimate the probability score for the success of the request and feed it to the final model.

**Final Model** The final model consists of the numeric data present in the dataset, the derived features based on previous works, and the probability estimates of the text from the previously mentioned text models performing well. We use the Logistic Regression [24], Gaussian Naive Bayes [25], and Random Forest [26] algorithms to create a classification model using the target variable of receiving a pizza as the dependent variable and the other dense features the independent variable. Our model handled unseen data without overfitting issues because we implemented a Stratified K-Fold cross-validation method [12].
Our findings with regards to following models are given under Section 4 with Accuracy [29], Precision [29], Recall [29], and F1-Score [29] as the metrics.

### 3.5 Parameter Optimization

Parameter Optimization or tuning is the name given to the process of selecting parameter values for a learning algorithm that leads to better performance of the algorithm. In this project, we decided to tune the hyperparameters of the Random Forest [26] algorithm, as there are a lot of parametric values that can affect the algorithm's performance. However, the parameters are not critical enough to need tuning for other algorithms, like Logistic Regression [24] and Naive Bayes [25]. This paper uses Randomized Search [30] to optimize hyperparameters by randomly selecting points assigned inside a domain. Following this, Grid Search [30] is applied to fine-tune the hyperparameters found by forming a grid of values based on the best values provided by the Randomized Search [30].

### 3.6 Feature Importance

Feature Importance is the technique of calculating a score for all the input features used in the model. This score refers to the importance of the variable in predicting the target variable. We use the built-in Feature Importance technique in the Random Forest algorithm implemented in scikit-learn using Gini Importance [31]. Gini Importance [31] averages the decrease in node impurity for each feature over all trees in the ensemble to get the feature importance. We also use the SHAP [32] interpretation to find the feature importances from the different models. In order to get this estimate of feature contribution to the prediction, it uses the Shapley [32] from game theory which helps us fairly distribute the contribution among the features. We apply both methods to all the final models with the self-created dense features as well as the probability estimates from the text models.

## 4 Results & Inference

To measure the performance and understand the feature importance in the parts mentioned before, we decided to divide the dataset of 5671 into 4537 training rows and 1134 testing rows. This helps expose the classifier to training on the given data and measure its performance over the unseen data.

### 4.1 Model Results

For the performance of the text models, we can infer from the above plot, developed using matplotlib, that out of the three machine learning algorithms used, Random Forest [26] performs equally well for each text embedding technique, while the performance under Logistic Regression [24] improves with the use of a more complex and advanced text embedding technique. However, Gaussian Naive Bayes [25] Classifier fails to achieve a reasonable threshold with any embedding techniques. Therefore, based on the inference from the graph, we choose the Random Forest version of all the embeddings.
Now to measure the performance of the final model, the test set separated from the primary data is first converted into vectors using the different embedding methods and then fed into the final model along with other numeric data. We make use of the metrics like Accuracy, Precision, Recall, and F-Score [29] in order to understand how the final model is performing on the test set.

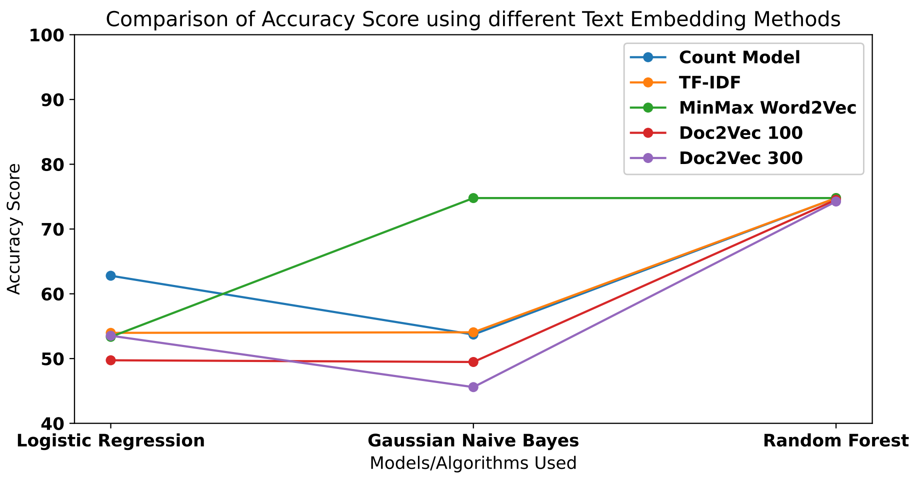

**Fig. 2.** Comparison of Accuracy Score using different Text Embedding Methods

Fig 3. displays metrics of different machine learning algorithms applied over the original numeric data combined with the estimated probability received from the Random Forest versions of the embedded text data. We see that the model's accuracy remains highest at around 75 when we use Random Forest to model the final data regardless of the embedding method used. However, for our use case, we are predicting whether a user will receive a pizza or not, which is a rare outcome with a chance of 24.63%, according to our dataset. Therefore, we would also look at the measure of recall we are getting from the different models. From the graph, we infer that both the Doc2Vec embeddings, i.e., 100- and 300-dimensions, fed into a logistic model perform much better than other models in predicting the requesters that will receive a pizza. However, we also notice that the same models are less precise in predicting who actually receives the pizza and who does not.

On the other hand, the Logistic Model of the final data having Count and TF-IDF embeddings have a comparatively lower but appropriate recall score as well as a good precision score. Therefore, we attempt to get a balance between recall and precision by measuring the F-Score of the models. We see that, on average, all the models perform relatively close to each other in terms of F-score, with the Logistic Regression version of Min-Max Word2Vec & TF-IDF and Random Forest version of 300-dimension of Doc2Vec & Min-Max Word2Vec have high F-scores. This suggests that the mentioned models maintain a good balance between precision and recall while predicting the outcome of a requester receiving a pizza or not.

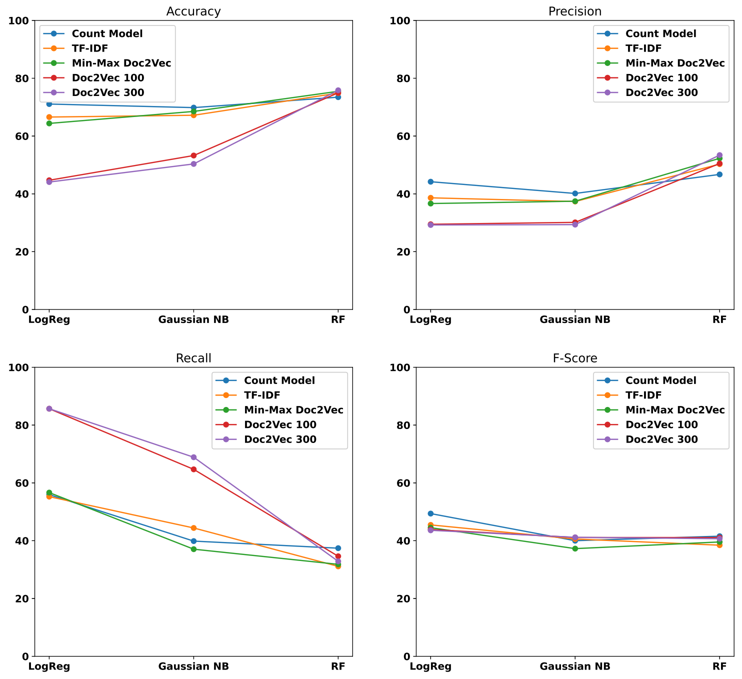

**Fig. 3.** Performance of the Final Model in terms of Accuracy, Precision, Recall & F-Score

### 4.2 Model Inference

SHAP [32] values from the classifiers provides the mean SHAP [32] values of each feature utilized to predict the target variable. This mean-SHAP [32] value refers to the impact of a feature on the model. Therefore, we see that the probability estimate generated from the text has a significant role in predicting whether a requester will receive their pizza or not. The number of comments on the request post, the upvotes minus downvotes, and the length of the text also have some part to play in predicting the outcome. However, from the models, we see that it hardly matters whether the requester has posted before in the subreddit or has attached an evidenced link in the text as their SHAP [32] values are very low.

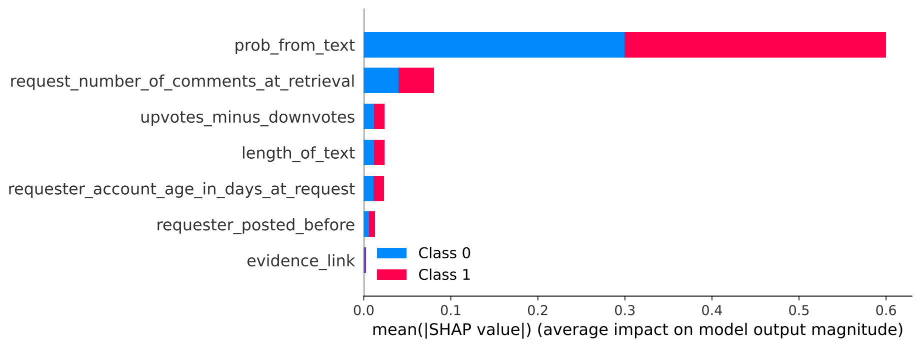

**Fig. 4.** Ranking of Feature Importance in terms of SHAP values

## 5 Conclusion

This work presents a stack of two models that separately deal with the text and numeric data. This allows us to reduce higher dimensional vectors generated from the text data to a probability estimate and retain the importance of the features present in the original dataset. The experimental findings validate the efficacy of our approach in terms of the performance metrics discussed in section 4. Therefore, given a requester posting a request, our model can predict whether it will stimulate any altruistic behavior from the other Redditors. Inferring the model also gives us an understanding of how useful other variables in the dataset were in predicting the outcome. In order to improve our work in the future, we can work around framing and combining different embedding techniques to get a vector representation of text, leading to better probability estimates and more satisfactory performance. Finally, this work can also be extended by using more sophisticated machine learning and deep learning models, as well as adding more data in order to develop a more robust model.

## References

1. Hutchins, W.J. (2004). The Georgetown-IBM Experiment Demonstrated in January 1954. In: Frederking, R.E., Taylor, K.B. (eds) Machine Translation: From Real Users to Research. AMTA 2004.
2. Tom Young, Devamanyu Hazarika, Soujanya Poria, Erik Cambria (2017). Recent Trends in Deep Learning Based Natural Language Processing. arXiv:1708.02709.
3. Bamman, D., Doğruöz, A. S., Eisenstein, J., Hovy, D., Jurgens, D., O'Connor, B., . . . Volkova (2016). Proceedings of the First Workshop on NLP and Computational Social Science.
4. Althoff, T., Salehi, N., Nguyen, T. (2013). Random Acts of Pizza : Success Factors of Online Requests.
5. Althoff, T., Salehi, N., Nguyen, T. (2013). Random Acts of Pizza : Success Factors of Online Requests.
6. J. Filipczuk, E. Pesce & S. Senatore, Sentiment detection for predicting altruistic behaviors in Social Web: A case study, 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016.
7. Hsieh, Hsun-Ping, Yan, Rui, Li, Cheng-Te. (2016). Will I Win Your Favor? Predicting the Success of Altruistic Requests. Pacific-Asia Conference on Knowledge Discovery and Data Mining.
8. Ahmad, Amreen & Ahmad, Tanvir  Bhatt, Abhishek. (2019). A Novel Approach for Predicting the Outcome of Request in RAOP Dataset. Proceedings of GUCON 2018.
9. Esin Durmus & Claire Cardie. 2018. Exploring the Role of Prior Beliefs for Argument Persuasion. In Proceedings of the 2018 Conference of the North American.
10. Diyi Yang, Jiaao Chen, Zichao Yang, Dan Jurafsky, and Eduard Hovy. 2019. Let's Make Your Request More Persuasive: Modeling Persuasive Strategies via Semi-Supervised Neural Nets on Crowdfunding Platforms. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.
11. Chen, J., & Yang, D. (2021). Weakly-Supervised Hierarchical Models for Predicting Persuasive Strategies in Good-faith Textual Requests. AAAI.
12. Raschka, S. (2018). Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning. CoRR, arXiv:1811.12808.
13. Loper, Edward & Bird, Steven. (2002). NLTK: the Natural Language Toolkit. CoRR, arXiv:cs/0205028
14. Erwig, Martin & Gopinath, Rahul. (2012). Explanations for Regular Expressions.
15. Fellbaum, C.D. (2000). WordNet : an electronic lexical database. Language, 76, 706.
16. Harris, David & Harris, Sarah. (2007). Digital Design and Computer Architecture.
17. Manning, C., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge: Cambridge University Press.
18. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). "Efficient Estimation of Word Representations in Vector Space." arXiv:1301.3781.
19. Le, Q. V., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents.
20. H. P. Luhn, A Statistical Approach to Mechanized Encoding and Searching of Literary Information, in IBM Journal of Research and Development.
21. SPARCK JONES, K. (1972), A Statistical Interpretation of Term Specificity and its Application in Retrieval, Journal of Documentation.

22. Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana (2020). Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems.
23. Boom, C. D., Van Canneyt, S., Demeester, T., & Dhoedt, B. (2016). "Representation learning for very short texts using weighted word embedding aggregation." Pattern Recognition Letters, 80, 150–156.
24. Cramer, J.S., The Origins of Logistic Regression (December 2002). Tinbergen Institute Working Paper No. 2002-119/4
25. Rish, Irina. (2001). An Empirical Study of the Naïve Bayes Classifier. IJCAI 2001 Work Empir Methods Artif Intell. 3.
26. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).
27. Cunningham, Padraig & Delany, Sarah. (2007). k-Nearest neighbour classifiers. Mult Classif Syst.
28. Evgeniou, Theodoros & Pontil, Massimiliano. (2001). Support Vector Machines: Theory and Applications.
29. Goutte, Cyril & Gaussier, Eric. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Lecture Notes in Computer Science.
30. James Bergstra, Yoshua Bengio. (2012). Random Search for Hyper-Parameter Optimization.
31. Menze, Bjoern & Kelm, Bernd & Masuch, Ralf & Himmelreich, Uwe & Bachert, Peter & Petrich, Wolfgang  Hamprecht, Fred. (2009). A comparison of Random Forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data.
32. Lundberg, S., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. arXiv:1705.07874