EasyChair Preprint

№ 5458

Identifying Legal Party Members from Legal Opinion Texts Using Natural Language Processing

Chamodi Samarawickrama, Melonie de Almeida,
Amal Shehan Perera, Nisansa de Silva and Gathika Ratnayaka

May 4, 2021

Identifying Legal Party Members from Legal Opinion Texts using Natural Language Processing

Anonymous submission

Abstract. Law and order is a field that can highly benefit from the contribution of Natural Language Processing (NLP) to its betterment. An area in which NLP can be of immense help is for information retrieval from legal documents which function as legal databases. The extraction of legal parties from the aforementioned legal documents can be identified as a task of high importance since it has a significant impact on the proceeding contemporary legal cases. This study proposes a novel deep learning methodology that can be effectively used to find a solution to the problem of identifying legal party members in legal documents. In addition to that, in this paper, we introduce a novel dataset which was created by an expert in the legal domain. Evaluations for the solution presented in the paper show that our system has 90.89% precision and 91.69% recall for an unseen paragraph from a legal document, thus conforming the success of our attempt.

Keywords: Legal party identification, Recurrent Neural Networks, Coreference resolution, NER

1 Introduction

Law and order is undoubtedly crucial for the proper functioning of society for without law there would be chaos, failing to offer equality to everyone. The legal domain being such a vital field, the incorporation of artificial intelligence into its work has drawn attention in many research works. This study is also one such endeavor taken towards building an automated legal system that eventually will be capable of extracting information from court cases and providing analysis and insights. The necessity of an automated legal system can be elaborated with a few major arguments, the first being the existence of case law. Case law by definition is the collection of previous judicial decisions that can be brought forth to clear the ambiguities in current cases. Therefore, the legal officials involved with a certain case are required to be knowledgeable about similar cases that have taken place before the ongoing one. This task is made tedious by the abundance of documents and records in the legal domain and the unavailability of a mechanism to perform intelligent queries on the said records. With a setting as such, it is apparent that an information extraction system for legal documents can be quite beneficial since it would help the legal officials in the course of gathering relevant information. In the process of extracting information, we observed that

the identification of legal parties in a case is crucial since a legal document is usually structured basing the said parties where the arguments and counter-arguments are presented concerning them. For this study, we have considered using legal opinion texts in terms of the dataset creation and testing. An opinion text is a statement written explaining the prevailing conditions of a case, which also contains background information at times where it is felt as required. By looking at an opinion text, one can get a sufficient understanding of the case and also of the factors leading to the decision of the case. Therefore, we believe opinion texts suits well in the attempt we take to reach a system that intelligently extracts information.

A legal party is any individual, a group of individuals or a body that can be held accountable(i.e. organization) for law and they are usually affected by or have an interest in the outcome of the legal case. Two major parties are identified in a legal case as the petitioner, that is the entity filing the case, and the defendant, the entity who is being prosecuted. Nevertheless, there are many complications when identifying legal parties in a document. As pointed out by Krass et al. in the Challenges Facing NLP in Legal Context[1], the unique hierarchical structure of the outcome, the linguistic quirk of legal adversarial, and the challenge of using acontextually trained embeddings are some of them. On top of that, we identify the ambiguity in the deciding of an entity to belong to a legal party or not also as an added hindrance to achieve our goal.

Example 1

- Sentence 1.1: *Petitioner Jae Lee moved to the United States from South Korea with his parents when he was 13.*
- Sentence 1.2: *During the plea process, Lee repeatedly asked his attorney whether he would face deportation; his attorney assured him that he would not be deported as a result of pleading guilty.*

Example 1 demonstrates the ambiguity of identifying the legal parties in a legal document. *his attorney* in Sentence 1.1 is identified to belong to a legal party in this case of *Jae Lee v. United States*[2] but *his parents* in Sentence 1.1 is not. This can only be understood after going through the entire document(or a sufficient portion of it) and comprehensively grasping the nature of the states. An intelligent way of processing the text is therefore needed and it is obvious that there is more intrinsic work that goes into identifying legal parties than the mere identification of people from a text.

2 Related Work

2.1 NLP in Legal Domain

Law and order is a domain that has drawn constant attention in the research areas of Natural Language Processing. Specific traits in the language used in this field have challenged the researchers and developers working on NLP tools to tackle the already existing defects in them and has constantly pushed them to level up their work. Ontology population[3, 4], discourse[5, 6] and semantic similarity[7, 8] are some areas where extensive work has been carried out concerning the legal domain.

Identifying Participant Mentions and Resolving Their Coreferences in Legal Court Judgements [9] is a work that is quite similar to our study where Gupta et al. have worked on resolving coreference for entities that belong to a party in legal texts. They have tried to address the issue of terms like *petitioner*, *defendant*, *appellant* not getting included in the coreference resolution by the already existing tools.

2.2 Coreference Resolution

Identifying mentions of the same entity in different forms and different positions in a text can be defined as coreference resolution in simple terms.

Example 2

- Sentence 2.1: *During the plea process, Lee repeatedly asked his attorney whether he would face deportation.*
- Sentence 2.2: *His attorney assured him that he would not be deported as a result of pleading guilty.*

The two sentences in Example 2 are taken out from the *Lee v. United States*[2] where the two sentences appear consecutively. The words *Lee, his, he* in Sentence 2.1 and the words *His, him, he* refer to Jae Lee, who is the petitioner in this case; and the term *his attorney* in both of the sentences is a different entity who appears as a defendant in this case. A mapping between these words is required to identify them as the same entity, rather referred to as tokens in Natural Language Processing(NLP) and coreference resolution delivers to that task.

Stanford and Spacy are two widely available and popular tools that offer coreference resolution, where both of the systems are built upon the following of Clark and Manning[10]. With an initial evaluation we conducted for both the systems for cases taken out from the domain we work on, we decided to carry our work with the Stanford system due to the better performance it showed[11].

2.3 Named Entity Recognition

The task of segregating entities into predefined categories is simply known as Named Entity Recognition(NER). These categories could be anything defined by the user, but in generic NLP tools that offer NER, the available common categories include classes such as PERSON, ORG, LOCATION, etc.
Spacy[12] and Stanford NER system[13] can be named as the two most popular tools for NER, and as similar to how we evaluated the coreference resolution, we evaluated these two systems with our cases and observed better performance in the Stanford system. Therefore, we have used the Stanford NER system for NER in this study.

2.4 Legal Entity Identification

Party Identification of Legal Documents using Co-reference Resolution and Named Entity Recognition[11] has conducted similar research on the same research problem where a rule-based approach to do the legal entity identification is presented. In this rule-based approach, coreference resolution is first performed and clusters of entities are identified in the document. Then, the identified clusters are filtered out where only the entities which are either a person or an organization are picked. Afterward, the number of times each entity is appearing as subjects in the text is taken into consideration when calculating the probability of each entity to be an actual legal entity.

2.5 Sequence to Sequence Learning for Legal Party Identification

This is a similar attempt to the Party Identification of Legal Documents using Co-reference Resolution and Named Entity Recognition[14] but with deep learning, approach to identifying the legal parties. A Recurrent Neural Network(RNN) encoder-decoder model with Long Short Term Memory(LSTM) cells is used in this study where masked sentences(a mask is applied to person/organization entities) are used as the input and gives an output sequence of 1s and 0s, 1 denoting the token is referred to a legal party in the document and 0 if otherwise. This model uses character-wise encoding to represent the words, that is assigning a real value to each unique character to form the word. We have further improved this approach by introducing word embeddings since we believe the vectors help to carry more meaning with the higher dimensionality it offers.

2.6 Recurrent Neural Networks

Recurrent Neural Network (RNN) is a class of artificial neural networks that are best suited for dealing with sequential data. RNN's ability to process inputs of variable length has helped it to show excellent performance in NLP related tasks [15]. Gated Recurrent Unit(GRU) [16] or Long Term Short Term Memory(LSTM) [17] cells can be used to retain the previous hidden state and the current input in RNNs and this study experiments with both of these approaches to find the best-suited architecture for its intended task. The obtained results for this are presented in the Experiments section.

2.7 Bidirectional Recurrent Neural Networks

Future input information is also important for prediction. This can be solved with RNN by delaying the output by a certain number of time steps to use future information for current prediction. But all future information cannot be captured. Bidirectional recurrent neural network (BRNN) that can be trained using all available input information in the past and future of a specific time frame has been proposed by Schuster et al.[18] to overcome the limitations of a regular RNN. The state neurons of a regular RNN are divided into two parts. One is responsible for the positive time direction and the other is for the negative time direction. Outputs from positive time direction and inputs of negative time direction are not connected, and vice versa. So that the BRNN can be trained with the same algorithms as a normal RNN.

2.8 Word Embedding

Word embedding is a technique used to represent words in the form of vectors while preserving their meaning. Many research studies in the field of law and order as well as other domains have been conducted with the incorporation of word embedding due to the fine performance this technology offers.

In this study, we have incorporated a pre-trained Word2Vec[19–21] model to create vector embedding of the tokens in the text we use to train our model. We use pre-trained vectors trained on part of Google News dataset where each word is represented by a vector of dimension 300[1].

3 Methodology

We discuss a novel method to accurately identify the members of the legal parties involved in a given legal case using deep learning. From here onward the members of the legal parties involved in the case are referred to as party members. Our method consists of four main steps as Tokenizing, Embedding, Masking, and Neural Network Model as shown in Figure 1.

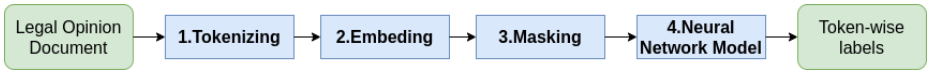

Fig. 1. Methodology

[1] https://code.google.com/archive/p/word2vec/

3.1 Tokenizing

First, the given court case paragraph is passed onto the Stanford annotator to split the text into a sequence of tokens and to get the Coreference Resolution and NER results of each token. Tokens can be either words, numbers, or punctuation marks. Out of these tokens, the entities that are either a PERSON, ORGANIZATION, or a LOCATION are identified with the use of Named Entity Recognition because only those entities can be a party member. Afterward, coreference resolution is performed on thus identified person/organization/location entities, and their corresponding mentions are identified to form groupings of the tokens to which in this paper we refer to as coref clusters. The NER value of the headword of each coref cluster is then passed down to the other tokens of the coref cluster and mapping is created to store the information token-wise (the fact that a certain token refers back to a person/organization/location entity) to generate masks later.

3.2 Embedding

In this step, a vector for each token that is of dimension 300 is generated with the use of a pre-trained model proposed by Google[2]. Before generating the vector, we also make sure to pass the token through a stemmer to increase the probability of the model containing the word. In a scenario where the model does not have an already trained vector for the token(This may happen for tokens such as numbers, punctuation marks, proper nouns etc.), a zero vector with the same dimension is returned as the corresponding vector for the token. We decided to use word2vec because we needed to get word-wise embedding by considering each word as an atomic entity.

3.3 Masking

In this step, an additional value (v) is generated for tokens that are identified as either a person or an organization, or a location. The logic that goes into deciding this value is explained with the Algorithm 1. The mask value (v) takes a value between 0 and 1 where the range is further divided into smaller ranges for PERSON, ORGANIZATION, and LOCATION classes. A PERSON entity gets a mask value v of range $0 < v < 0.5$, an ORGANIZATION entity gets a mask value v of range $0.5 < v < 0.75$ and a LOCATION entity gets a mask value v of range $0.75 < v < 1.0$. The reason we assigned a wider range to the PERSON entities in comparison with the other two is that we observed in opinion texts PERSON entities appear in higher frequencies than the other two. All the tokens of the same coref cluster identified in the tokenizing step are given the same value. The vector of each token given by the previous step are masked or extended with values generated for each token by Algorithm 1.

[2] https://code.google.com/archive/p/word2vec/

Example 3

– The Federal Trade Commission (FTC) filed an administrative complaint, alleging that the Board's concerted action to exclude non dentists from the market for teeth whitening services in North Carolina constituted an anti competitive and unfair method of competition under the Federal Trade Commission Act.

In the Example 3 "The Federal Trade Commission (FTC)" and "the Board" are ORGANIZATION entities and "North Carolina" is a LOCATION entity. So that the tokens: "The", "Federal", "Trade", "Commission","(", "FTC" and ")" are masked with a value $v_1(0.5 < v_1 < 0.75)$, the tokens: "the" and "Board" are masked with a value $v_2(0.5 < v_2 < 0.75)$ and the tokens: "North" and "Carolina" are masked with a value $v_3(0.75 < v_3 < 1.0)$.

Algorithm 1 Mask Value Generator

Input: $corefs, NERvaluesofheadwordsofclusters$
Output: $D_{maskValues}$

procedure ValGenerator($corefs$)
  for each $cluster_k$ **in corefs do**
    $head_k \longleftarrow cluster_k.headword$
    if $head_k.NER == PERSON$ **then**
      $val_k \longleftarrow v_1(0 < v_1 < 0.5)$
    end if
    if $head_k.NER == ORGANIZATION$ **then**
      $val_k \longleftarrow v_2(0.5 < v_2 < 7.5$
    end if
    if $head_k.NER == LOCATION$ **then**
      $val_k \longleftarrow v_2(0.5 < v_2 < 1.0$
    end if
    if $head_k.NER! = PERSONorORGANIZATIONorLOCATION$ **then**
      $val_k \longleftarrow 0$
    end if
  end for
  $D_{maskValues} \longleftarrow setOf(head_k : val_k)$
return $D_{maskValues}$

end procedure

3.4 Neural Network Model

The input of this model is the sequence of masked vectors $\{X_1, ..., X_T\}$ of the given paragraph and the output is a sequence of probabilities $\{y_1, ..., y_T\}$, where

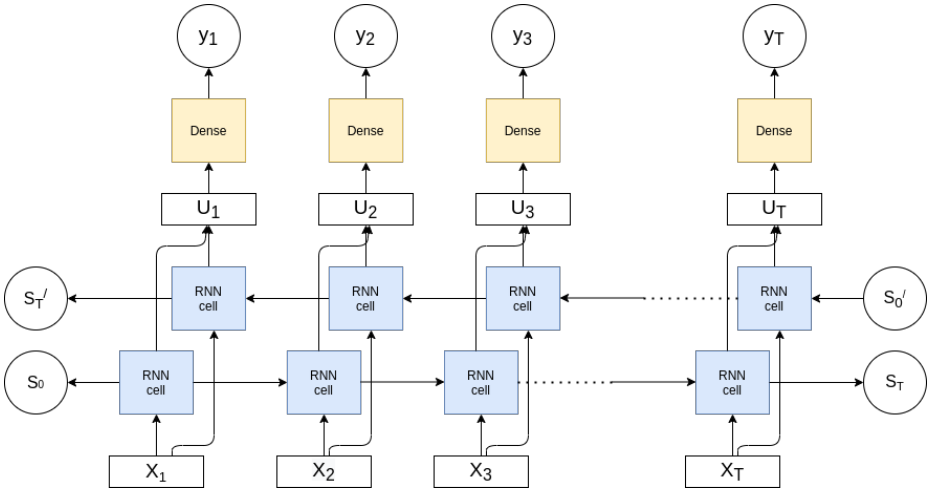

Fig. 2. Architecture of the Neural Network Model

T is the number of tokens in the paragraph. If a token ($token_t$) of the given paragraph is referred to as a party member then the output value (y_t) corresponding to that token must be close to 1 and otherwise it must be close to 0. Figure 2 depicts the architecture of the model we designed for this task. We feed the sequence of masked vectors $\{X_1, ..., X_T\}$ into a sequence of BRNN cells (BRNN layer). We use BRNN instead of regular RNN because the model needs all the information about the given text to decide whether any token is referred to a party member or not. The sequence of output vectors generated by the BRNN layer $\{U_1, ..., U_T\}$ is fed into dense layers to generate the probability of each token to be referred to a party member. We perform a token-wise binary classification at the output layer of the model using the sigmoid activation function.
In Example 3, "The Federal Trade Commission (FTC)" and "the Board" are two party members involved in the case. So that the tokens: "The", "Federal", "Trade", "Commission","(", "FTC", ")", "the" and "Board" are referred to party members. Therefore outputs corresponding to those tokens need to be 1, and outputs corresponding to all the other tokens need to be 0.

Training Phase In this phase, Neural Network Model is trained to minimize the Binary Cross-Entropy using a dataset that consists of a 3D input array of size (m,n,T) and a 2D expected output array of size (m,T). 3D input array contains a set of sequences of vectors. 2D expected output array contains the corresponding set of sequence of binary values (1s and 0s). In this phase, the model basically learns parameters of BRNN and Dense layers to identify tokens of a text sample that are referred to a party member.
m = number of sample paragraphs

n = dimension of a token vector after masking(301)
T = maximum possible number of tokens in a given sample paragraph
If the number of tokens in a given sample paragraph is less than T, zero vectors of size n are added to fill the 3D array.

Inference Phase In this phase, the trained Neural Network Model takes the masked word embedding of the given paragraph which is an array of size (n,T), and gives the corresponding result vector of size T. Each value of the result vector is between 0.0 and 1.0 which is the probability of each token to be referred to a legal party of the given paragraph. In this phase also, if the number of tokens in a given paragraph is less than T, zero vectors of size n are added to fill the array. In this phase, the model uses learned parameters of BRNN and Dense layers to identify tokens of a text sample that are referred to as a party member.

4 Experiments

4.1 Setup

Natural Language Software: Stanford CoreNLP tools were used to perform NER and coreference resolution and for stemming purposes, the Porter stemmer by nltk[22, 23] was used.
Deep Learning Software: We used Keras[24] open source library which runs on top of Tensorflow python library to implement the Neural Network Model
Word Embedding Software: We used a pretrained word2vec model by Google to generate vectors in the Embedding step.

4.2 Dataset

We created a list of 1000 sample paragraphs (samples) that are picked from legal opinion texts and tokenized each sample using the tokenizing method we explained in the methodology. Then we manually labeled each token of each sample as a party member mentioned in each sample or not with the help of an expert of the legal domain. Then we generated masked word embedding of each sample by following the embedding and masking steps. The statistics of our dataset is as mentioned in the Table 1. The number of tokens that are referred to as party members are just about 3.46% of the total number of tokens of a paragraph.

Table 1. Statistics of the Dataset

Attribute	Train	Val	Test	Fullset
Cases	810	90	100	1000
Tokens	351K	39K	43K	433k
Party member tokens	12.16K	1.21K	1.62K	14.99K

4.3 Performance of the Neural Network Model

This model was trained for 100 epochs with a learning rate of 0.01 separately with Gated Recurrent Units (GRU) and Long short-term memory (LSTM) with alterations done to the number of output units of the BRNN layer. We used Binary Cross Entropy as the loss function with Adam Optimizer. We used Precision at the given Recall as the metric to train the model because according to Table 1, the number of expected positives is much less than the number of expected negatives. Performance of the model according to accuracy, precision, recall, f1 score, and the average training time per step is shown in Table 2. Equations 1 2, 3 are the definitions of Accuracy, Precision, Recall and F1 score.

TP - True Positives, TN - True Negatives, FP - False Positives, FN - False Negatives

$$A = (TP + TN)/(TP + TN + FP + FN) \tag{1}$$

$$P = TP/(TP + FP) \tag{2}$$

$$R = TP/(TP + FN) \tag{3}$$

$$F1score = 2 * P * R/(P + R) \tag{4}$$

Training times shown in this table 2 are according to the performance of the Intel Xeon Processor with two cores @ 2.30 GHz and 13GB RAM. We can see that as the number of output units increases, the Accuracy, Precision, Recall, and F1 score of models with both GRU and LSTM has increased. The model with GRU cells of 512 output units has shown the best performance. Also, the time consumption of GRU is much less than LSTM when the complexity of the model increases. These results are a clear indication of the accuracy of our methodology to identify legal party members.

Table 2. Performance of the Neural Network Model

		Performance				
BRNN cell type	Output units	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)	Training time per step (s)
GRU	8	97.74	70.38	44.78	54.73	0.42
	32	98.41	78.95	58.15	66.97	0.63
	64	98.95	79.80	73.06	76.28	1.00
	128	99.10	79.49	79.21	79.35	2.00
	256	99.56	86.46	86.56	86.51	4.00
	512	99.88	90.89	91.69	91.29	15.0
LSTM	8	97.58	67.77	34.58	45.79	0.43
	32	98.37	74.77	58.99	65.95	0.73
	64	98.72	81.28	63.90	71.55	2.00
	128	99.34	85.02	79.71	82.28	3.00
	256	99.52	87.50	84.30	85.87	7.00
	512	99.82	89.70	92.16	90.91	20.0

5 Conclusion

We propose a natural language processing method to accurately predict the members of legal parties, given a paragraph of a legal opinion text. First, We identify the entities that are either a person or an organization or a location and spot their mentions in the paragraphs. Our model then proceeds to evaluate the likelihood of each such mention to be referred to as a legal party member by inspecting the meaning of the paragraph using BRNN. The meaning is grasped by the model with the help of word embedding and learned with the training process. Also, we introduced a dataset that can be used to train our model. We show that our system has 90.89% precision and 91.69% recall for an unseen paragraph from a legal document. So this method can be used to identify the entities that are most likely to be a legal party.

References

1. M. S. Krass, "Learning the rulebook: Challenges facing nlp in legal contexts."
2. "Lee v. United States," in *US*, vol. 432, no. No. 76-5187. Supreme Court, 1977, p. 23.
3. V. Jayawardana, D. Lakmal, N. de Silva, A. S. Perera, K. Sugathadasa, and B. Ayesha, "Deriving a representative vector for ontology classes with instance word vector embeddings," in *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*. IEEE, 2017, pp. 79–84.
4. V. Jayawardana, D. Lakmal, N. de Silva, A. S. Perera, K. Sugathadasa, B. Ayesha, and M. Perera, "Word vector embeddings and domain specific semantic based semi-supervised ontology instance population," *International Journal on Advances in ICT for Emerging Regions*, vol. 10, no. 1, p. 1, 2017.
5. G. Ratnayaka, T. Rupasinghe, N. de Silva, M. Warushavithana, V. Gamage, and A. S. Perera, "Identifying relationships among sentences in court case transcripts using discourse relations," in *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, 2018, pp. 13–20.
6. G. Ratnayaka, T. Rupasinghe, N. de Silva, M. Warushavithana, V. S. Gamage, M. Perera, and A. S. Perera, "Classifying sentences in court case transcripts using discourse and argumentative properties," *ICTer*, vol. 12, no. 1, 2019.
7. K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera, "Synergistic union of word2vec and lexicon for domain specific semantic similarity," in *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*. IEEE, 2017, pp. 1–6.
8. ——, "Legal document retrieval using document vector embeddings and deep learning," in *Science and information conference*. Springer, 2018, pp. 160–175.
9. A. Gupta, D. Verma, S. Pawar, S. Patil, S. Hingmire, G. K. Palshikar, and P. Bhattacharyya, "Identifying participant mentions and resolving their coreferences in legal court judgements," in *International Conference on Text, Speech, and Dialogue*. Springer, 2018, pp. 153–162.
10. K. Clark and C. D. Manning, "Deep reinforcement learning for mention-ranking coreference models," *arXiv preprint arXiv:1609.08667*, 2016.
11. Anonymous, "Party identification of legal documents using co-reference resolution and named entity recognition."

12. M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, vol. 7, no. 1, 2017.
13. J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *ACL*, 2005, pp. 363–370.
14. Anonymous, "Legal party extraction from legal opinion text with sequence to sequence learning."
15. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.
16. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
17. A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.
18. M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
19. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.
20. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
21. T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2013, pp. 746–751.
22. S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.
23. E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.
24. F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.