Juris2vec: Building Word Embeddings from Philippine Jurisprudence

Elmer Peramo
Advanced Science and Technology Institute (ASTI)
Department of Science and Technology (DOST)
Quezon City, Philippines
elmer@asti.dost.gov.ph

Charibeth Cheng
College of Computer Studies
De La Salle University
Manila, Philippines
charibeth.cheng@dlsu.edu.ph

Macario Cordel II
College of Computer Studies
De La Salle University
Manila, Philippines
macario.cordel@dlsu.edu.ph

2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) | 978-1-7281-7638-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICAIIC51459.2021.9415251

***Abstract*— In this research, we trained nine word embedding models on a large corpus containing Philippine Supreme Court decisions, resolutions, and opinions from 1901 through 2020. We evaluated their performance in terms of accuracy on a customized 4,510-question word analogy test set in seven syntactic and semantic categories. Word2vec models fared better on semantic evaluators while fastText models were more impressive on syntactic evaluators. We also compared our word vector models to another trained on a large legal corpus from other countries.**

***Keywords*— natural language processing, word embeddings, word2vec, gloVe, fastText, intrinsic evaluation**

I. Introduction

A. Background

As in any other applications of machine learning, certain properties or features of data are represented as sets of numbers or vectors to facilitate further processing and analytics. Text in natural language processing (NLP) is no different, and the process of converting text into an ordered set of numbers is called text vectorization. One such representation is called one-hot encoding where each word is represented by a long but sparse vector containing a value of one in one spot and the rest are zeroes. That spot reflects the position of that word in a certain dictionary containing all the unique words taken from a corpus. Effectively, the length of the one-hot encoded vector is the same as the number of elements in that dictionary. Another method of representing a word is by counting how frequent it appears in a certain document resulting in what is called a term frequency (tf) vector. However, some words in the English language appear very commonly but do not contain a lot of information like the words *the*, *of*, and *it*. Because of this, tf-idf was formulated to give weights to important words in a document [1]. It is a statistical measure that factors in not just the term frequency but also the inverse document frequency. One-hot encoding and tf-idf are very common text representation techniques in NLP applications like sentiment analysis, classification, information retrieval, and topic modeling.

But perhaps as a result of the rapid growth of deep learning technology, word embeddings became one of the innovations in natural language processing. They are mappings of text to a vector space. Each piece of text or token (usually a word) are converted into a dense vector embedded in a dimension lower than what they would be if represented using the traditional one-hot encoding. Unlike the one-hot encoding representation of a word, which cannot encapsulate the similarity between words, they somewhat magically capture not just syntactic relations but also semantic associations of words. Word embeddings are sometimes called distributed representation.

Machine learning algorithms such as singular value decomposition and neural networks are commonly used to compute real-number values for each resulting vector in word embeddings. They can be automatically learned without the need for supervised learning or labeling. All that is needed are a large corpus, machine learning algorithms coded in a programming language, and a GPU-enabled machine.

Arguably, the three most popular word embeddings are word2vec, gloVe, and fastText, which we utilized in this research.

Word2vec generates word vectors by training a shallow neural network (two layers) where one-hot encoded vectors of size V (the length of the vocabulary) are fed into the input layer, which are linearly transformed to h intermediate nodes (dimensions). Then, a second fully-connected layer projects the h-dimensional vector into V output neurons in order to map the input word in a corresponding word presented in the same window. The output vectors are then converted into probability vectors using and efficient implementation of the softmax function. Two popular implementations of Word2vec are the continuous bag-of-words (CBOW) and the skip-gram model. The CBOW model tries to predict the target word (or the center word) based on the word in the context window, while the skip-gram model does the opposite by predicting the context words given the target word [2].

GloVe uses global (corpus-wide) statistical information in the form of a co-occurrence matrix [3]. Given a context window of size *x*, the number of times two words co-occur at that point in time is tallied in a term co-occurrence matrix. The logarithms of the counts are computed, and entries with zero values are removed. This large matrix containing the logarithm of the co-occurrence counts is then factorized using singular value decomposition (SVD) and applied a weighting function to relax the effect of extremely common or extremely rare co-occurrences. GloVe uses a weighted least squares optimization model in coming up with the optimal parameters.

A more recently introduced fastText model makes use of the subword information. The algorithm is basically the same as the skip-gram version of word2vec, except that it breaks down a word into character *n*-grams and compute a vector representation of each character *n*-grams using the word2vec algorithm. Ultimately, the vector representation of the word is computed as the sum of the vector representations of the character *n*-grams comprising that word [4].

The succeeding paragraphs discussed the problem statement, significance, objectives, the scope and delimitation of this research. Section II provides a discussion of some related literature including how this research is differentiated from or is complemented with these works. Section III characterizes the dataset and the processing involve in preparation for the model training. In this paper, we utilized three implementations of each of the above-mentioned algorithms. Section IV discusses the methodology, models, architectures, and hyperparameters used in the experiment including metrics for evaluation. Sections V discusses the results. Lastly, the conclusion and future work are discussed in Section VI.

B. *Problem Statement*

Reducing the backlog of cases in trial courts has long been the problem of the Philippine judiciary for so many years now. Automating some administrative processes would somehow accelerate the reduction of these cases. Automation will involve digitization of these documents, but currently, most of the documents being used in the Philippine legal system have not yet been digitized. And if they were, they are mostly stored or archived in some sort of monolithic and siloed systems. For these digitized documents, a powerful text representation would facilitate intelligent information retrieval, document classification, and deeper conceptual understanding. Properly trained word embeddings on legal documents can provide efficiency in eliciting clarity and unequivocality of concepts in these documents, thereby helping lawyers craft better strategy and judges come up with prudent decisions. As far as the researcher knows, there are no publicly available word embeddings trained on large legal corpora in the Philippines. Applying AI, deep learning, and natural language processing with the law will generally strengthen legal research, such that automated analysis based on the words and phrases (including their context) relations from laws and jurisprudence will be applied to 'teach' the computer system to predict or suggest outcomes of new cases, among other use cases. Word embedding generation is just the start, but this will lay the foundation for a more sophisticated natural language processing technique later.

C. *Objectives*

It is the hope of this research to improve information extraction and retrieval techniques in the judiciary and advance legal research in the country. Specifically, we aim to leverage the available data on Philippine jurisprudence and come up with 300-dimensional word embeddings that would be able to capture relationships between legal terms, elicit insight from various visualizations of the generated embeddings, potentially reveal semantic shifts in some legal words over time and bias (gender, racial, etc.) in the data source. We would also create a comprehensive list of word analogy test to intrinsically evaluate the performance of the generated embeddings.

D. *Significance of the Study*

Language is paramount to the study of law. Understanding the meaning and interpretations of its components can play a significant role in advancing the field of legal research at an accelerated pace comparable to how deep learning has made remarkable headway in the area of computer vision. These word embeddings—and the meaning embedded within the distribution of weights—have been employed extensively in computer science applications. But for legal domain, they can potentially give way to practical applications like legal document classification, information retrieval, word/phrase analogy, translation (e.g., language translation and paraphrasing of text in legalese into layman's language), question-answering, and legal case/briefs automatic summarization.

This could help judges and legal scholars have a deeper understanding and interpretation of jurisprudence. It could provide social scientists and historians insights into the temporal evolution of word meanings in the Philippine legal context.

This is an active research area in NLP with cascades of extensions and variation. Among the promising avenues of future research in this direction include exploring the temporal evolution of certain words in the embedding, debiasing techniques, dealing with polysemous words, and generating specific dictionaries of ideologies or schools of thoughts in judicial opinions.

E. *Scope and Delimitation*

We limited our corpus to the Supreme court decisions from 1901 to 2020 only and did not include legislation, executive orders, and the constitution. There are still no perfect evaluation methods testing the quality of the generated word vectors for linguistic relationships because it is difficult to understand exactly how the embedding spaces encode linguistic relations. In this case, intrinsic evaluations will exclude out-of-vocabulary (OOV) words.

II. RELATED WORK

Generic pre-trained word vectors based on word2vec, gloVe or fastText algorithms abound but most of them were trained on conventional encyclopedic corpora like Wikipedia and online articles, and news stories.

As of late, numerous specific-domain embeddings have been explored. The researchers were motivated and inspired by potentially impactful findings from these domain-specific embeddings that can capture rich properties of concepts and relationships among terminologies:

Tshitoyan et al. trained an unsupervised model from materials science knowledge bases present in the published literature. They were able to efficiently encode chemical information as dense word embeddings. Without any explicit

inclusion of knowledge chemical properties, these embeddings capture complex materials science concepts such as the underlying structure of the periodic table and structure–property relationships in matters. They were able to show that an unsupervised method and a large collection of chemistry information-rich documents can recommend materials for functional applications several years before their discovery [5].

Risch and Krestel used the power of word embeddings to examine the novelty of patent applications using automatic classification. The authors trained their model on more than five million patent documents [6].

Zhang et al., created BioWordVec, a distributed word representation that provided an essential foundation for biomedical natural language processing (BioNLP), text mining, and information retrieval. It is an open collection of biomedical word vectors/embeddings that combines subword information from unlabeled biomedical text using a widely used biomedical controlled vocabulary called Medical Subject Headings (MeSH). They have also assessed both the validity and the utility of their generated word embeddings over multiple NLP tasks in the biomedical domain [7].

Law2Vec is a legal word embedding created by Chalkidis (2018) containing a large number of legal corpora from various public legal documents in English. The collection comprises the following number of documents: 53, 000 UK legislation (e.g. UK Public General Acts, Local Acts, etc.); 62,000 European legislation (e.g. EU Treaties, Regulations, Directives, etc.) published in Eur-Lex; 5,500 Canadian legislation (e.g. Consolidated Acts, Constitutional Documents, etc.); 1,150 Australian legislation; 800 English-translated legislation from EU countries (e.g. Finland, Sweden, France, Germany, etc.); 780 English-translated Legislation from Japanese; 68 bound volumes of the US Supreme Court decisions from 1998 to 2017; and 54 titles of the most recently updated U.S. Code [8].

Figure 1. Visualization of some words in (Law2Vec)

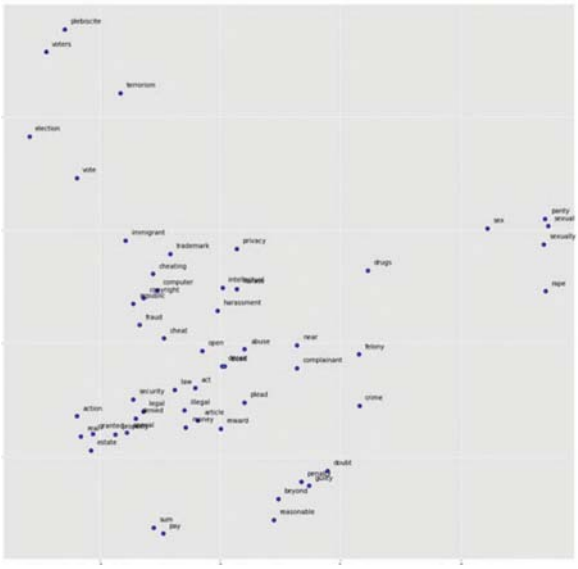

Figure 2. Visualization of some words in Juris2vec

Overall, the corpus includes in total 123,066 documents which consist of 492M individual words (tokens) and trained word2vec models to generate the word vectors. He trained two individual word2vec models for 100-dimensional and 200-dimensional embeddings using the gensim library. So far, this is the most relevant work that we can compare our results with. Figures 1 and 2 below show a visualization of how our model compare with the work of Chalkidis. Juris2vec's 300-dimensional word vectors and law2vec's 200-dimensional word vectors were reduced to two dimensions using principal component analysis (PCA) and plotted on a coordinate system.

Law2vec's corpus was more diverse, but it was trained only on 200-dimensional embedding space and using word2vec. Juris2vec, on the other hand, concentrates on jurisprudence documents, an exhaustive collection of supreme court decisions in the Philippines spanning more than a century. Juris2vec also made use of more sophisticated algorithms like gloVe and fastText, besides word2vec.

III. THE DATASET

The Philippine Jurisprudence is a collection of text containing Supreme Court decisions, resolutions, and opinions on cases escalated from the lower courts. They are written mostly in the English language, but there are occasional texts in Filipino and Spanish. Obviously, the body of text is also replete with legal jargons, which are mostly in Latin. The original documents from the Supreme Court were converted into an image using optical character recognition (OCR) for archiving purposes. Organizations like Arellano University School of Law (lawphil.net) and Chan Robles Law Firm (chanrobles.com) compiled the files based on the imaged document and maintain an online repository of the full text of this jurisprudence from as early as 1901 to as recent as 2020 and made them publicly available. However, these texts, due to the reconversion process, contain a lot of typos, misspellings, wrong formatting, and other structural misprints. The researchers took advantage of the free dataset and downloaded the "dirty" text online using a python web scraper script. Further text preprocessing and "cleaning" was done to the data like removing typographical errors, HTML tags, and CSS codes, unnecessary headers and footers, non-UTF8 characters, numerals, and words appearing less than 5

times. The text was split into lists of sentences using the NLTK library to provide the best possible input for the models. For consistency, all the words in the corpus were lower-cased. Punctuation marks and special characters were removed, and we constructed a matrix of co-occurrence counts for words appearing within a window of 20 words. Unnecessary headers and footers were also removed to facilitate text processing later on. We obtained a total of 60,485 documents (text files), which were merged into one big, cleaned corpus consisting of 196 million individual words (tokens) amounting to 1.1 Gigabytes in terms of file size.

FIGURE 3. TOP 25 MOST FREQUENT WORDS IN THE CORPUS

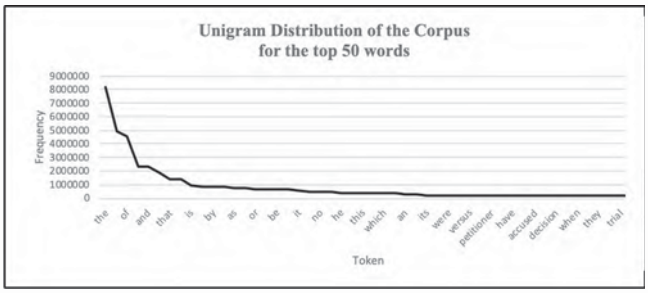

Figure 3. Top 25 most frequent words exhibiting Zipf's Law, typical of a large corpus.

Figure 3 offers an overview of the unigram distribution of words in the processed corpus showing just the top 25 words. The vocabulary, however, contains 183,942 unique tokens. It is easy to see that it exhibits a Zipfian behavior. As would be expected, stop words dominate the ranking and the distribution peters out after the word *it* with frequencies below 1,000,000.

As points of comparison, recent work in computational linguistics evaluating methods for extracting word embeddings utilized a 2010 Wikipedia dump with 1 billion tokens; a 2014 Wikipedia dump with 1.6 billion tokens; Gigaword 5 which has 4.3 billion tokens; the combination Gigaword5 + Wikipedia2014, which has 6 billion tokens; and 42 billion tokens of web data from Common Crawl. Taken together, the corpus of judicial opinions compares favorably to the work in computational linguistics employing these tools, with two of the common corpora actually featuring fewer tokens than the compiled judicial corpus. It is worth emphasizing the scale of the data is comparable to that utilized in prominent computer science applications, including the original GloVe article.

Table 1. Top-5 similar words for a set of 20 selected words based on cosine similarity using the word2vec model with context window = 15.

TABLE I. TOP 5 SIMILAR WORDS

Word	Nearest Neighbors
article	code, art, provisions, paragraph, section
act	acts, done, cannot, mere, clearly
action	suit, cause, complaint, instituted, actions
crime	murder, accused, offense, charged, committed
felony	conspiracy, crime, commit, felonious, crimes
penalty	reclusion, imposable, perpetua, imposed, penalties
security	guards, guard, tenure, secured, FEMJEG
fraud	fraudulent, deceit, misrepresentation, mistake, fraudulently
privacy	prying, expectation, individual's, privacies, confidentiality
intellectual	cognitive, mental, retardation, adaptive, discernment
election	elections, candidates, candidate, precinct, votes
immigrant	immigration, immigrants, alien, deported, deportation
illegal	illegally, dismissal, unlawful, recruitment, violation
drugs	dangerous, drug, shabu, marijuana, paraphernalia
appeal	appealed, appeals, decision, motion, certiorari
money	amount, payment, paid, cash, sum
alcohol	denatured, ethyl, spirits, alcoholic, distilled
complaint	alleged, action, filed, answer, against
indictment	indicted, information, indictments, offense, jeopardy
motion	reconsideration, motions, dismiss, filed, petition

IV. THE EXPERIMENTS

The researchers cloned the original C/C++ implementations of word2vec, gloVe, and fastText from their respective github pages. In the interest of time, we used two computers for training and model generation: a 2015 model MacBook Pro (8GB RAM/CPU/2 cores) and an Acer Aspire (Intel Core i7 CPU at 2.2 GHz/8 cores) running Ubuntu 16. In all the experiments, we just used a thread value of 2 to maximize CPU processing. We set the vector size to 300, which is the most common word embedding dimension used for large corpora. We design the experiments such that for each algorithm, we set a symmetric window of sizes 5, 9, and 15. These are ideal sizes with the given amount of data that we have. We also set a minimum count of 5 for all the models. This means that a word from the corpus occurring less than 5 times is removed from the vocabulary. This effectively reduces the size of the vocabulary to 183,942.

TABLE II. DESIGN OF EXPERIMENTS

Experiment ID	Algorithm	Loss Function	Window Size	Training Time[a]
1	Word2Vec - CBOW	Cross-entropy	5	454
2	Word2Vec - CBOW	Cross-entropy	9	114
3	Word2Vec - CBOW	Cross-entropy	15	65
4	Word2Vec – Skip-Gram	Negative Sampling	5	669
5	Word2Vec – Skip-Gram	Negative Sampling	9	331
6	Word2Vec – Skip-Gram	Negative Sampling	15	476
7	GloVe	log-bilinear regression	5	111
8	GloVe	log-bilinear regression	9	137
9	GloVe	log-bilinear regression	15	144
10	FastText	Negative Sampling	5	322
11	FastText	Negative Sampling	9	741
12	FastText	Negative Sampling	15	1,013

[a] Training time is in minutes.

V. RESULTS AND DISCUSSION

For each algorithm, epochs or the number of training iterations is set to 15 on the same set of words. For GloVe, models were trained using AdaGrad, and we set the value of alpha to 0.75 and max_iter to 100, which are the hyperparameter settings used in the original GloVe model. For fastText that leverages on character *n*-grams, we used a minimum value of 3 and a maximum value of 6 for *n*. We used an adaptive learning rate with an initial value of 0.05. For models that used negative sampling, we set the value of *k* to 5 (with a sampling threshold of 0.0001), which means that for every target word we match 5 negative samples in the training. The rest of the hyperparameters were set to their default value. Table II below details the experimental design that the researchers implemented. Because of the difference in approach of the algorithms presented here, the researchers tried their best to make each of the resulting models as comparable as possible.

Word2vec perform better at semantic category while fastText perform better on syntactical category. Generally, the implementation with longer window size performs better in the analogy test in terms of accuracy.

TABLE III. PERFORMANCE OF THE MODELS IN TERMS OF ACCURACY

Algorithm	Window Size	Ph. Geography-500		Gender Test-380		Opposites Test-505		Comparative-533		Participles-332		Adverb-812		Plural-1550		Overall Accuracy (4510)
		#No.	%	#No.	%	#No.	%	#No.	%	#No.	%	#No.	%	#No.	%	
Word2Vec	5	392	78%	233	56%	62	12%	46	9%	54	25%	32	4%	299	26%	27%
Word2Vec	9	386	77%	225	59%	57	11%	21	4%	51	23%	42	5%	338	22%	25%
Word2Vec	15	390	78%	207	54%	57	11%	24	5%	66	20%	77	9%	350	23%	26%
Glove	5	164	33%	149	47%	10	2%	14	3%	18	8%	16	2%	142	9%	12%
Glove	9	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0%
Glove	15	165	33%	156	41%	12	2%	21	4%	19	8%	12	1%	133	10%	11%
Fastext	5	274	55%	133	35%	103	20%	83	16%	78	24%	281	35%	871	56%	40%
Fastext	9	304	61%	115	30%	78	15%	62	12%	71	21%	322	40%	781	50%	39%
Fastext	15	312	62%	89	23%	74	15%	54	10%	80	24%	347	43%	721	47%	37%

The table above shows the number of items each Juris2vec word embedding model got correctly from various word analogy tests. The researchers concocted a total of 4510-word analogy tests on topics ranging from semantic tests of Philippine geography to a more syntactically attuned test on plural nouns and verbs. The Gender Test contains analogies of gender counterparts for each noun (e.g., 'man' is to 'woman' as 'king' is to 'queen'). Opposites Test contain items like 'honest' is to 'dishonest' as 'healthy' is to 'unhealthy'. Comparative Test includes items like 'big' is to 'bigger' as 'small' is to 'smaller'. Participles Test includes items like 'dance' is to 'dancing' as 'sing' is to 'singing'. The Adverb Test basically just contains analogies of adjectives converted to adverbs by adding 'ly' after the word, and the Plural Test by adding 's' or 'es' after the word.

The researchers assume that if more training time (i.e., more epochs or iterations) is given, the accuracy of any model would increase. Other factors that would improve the performance are the window size and the length (dimension) of the word vectors.

VI. CONCLUSION AND FUTURE WORK

In this work, we presented a domain-specific word embedding model called Juris2Vec, which we would also make publicly available for use in future experiments. The language of interest is in English, Spanish and Filipino. We trained Juris2vec from a large number of the legal corpus of Philippine Supreme Court decisions from 1901 to 2020. We opted to train based on the word2vec skip-gram model, gloVe and fastText. We were able to verify that fastText is better at syntactical evaluations because of the subword information embedded in each word vectors, whereas word2vec fare better at semantic evaluations.

Overall, the field of neural word embeddings is fascinating. Not only is the ability to mathematically capture semantic context and word relations academically intriguing, word embeddings have also been a hugely important driver behind many However, word embeddings are not without limitations, and ML practitioners sometimes turn to newer pre-trained language modelling techniques (e.g., ELMo, BERT, and OpenAI's generative pre-trained transformers) to overcome some of the inherent problems with word embeddings like polysemy, explainability, and OOV. Nevertheless, word embeddings remain one of the most fascinating NLP topics today, and the move from sparse, frequency-based vector representations to denser semantically representative vectors is a crucial step in advancing the NLP subdomain and the field of legal AI.

For future work, the researchers would like to explore other evaluation methods (e.g., extrinsic), other evaluation metrics (e.g., perplexity), the evolution of word thru time (e.g., per decades or scores) and how bias (racial, gender, or) can manifest in text using word embeddings. The extrinsic evaluation using the generated word embeddings as features in a document classification task classifier to use: Naive Bayes, Logistic Regression, Support Vector Machines and Random Forest (packages in scikit-learn) and comparing the result with document classification using traditional NLP (n-grams, tf-idf, bag-of-words) using NLTK and spaCy.

REFERENCES

[1] K.S. Jones, "A statistical interpretation of term specificity and its application in retrieval," Journal of Documentation, ISSN: 0022-0418, 1 January 1972.

[2] T. Mikolov et al., "Efficient estimation of word representations in vector space", in International Conference on Learning Representations, 2013.

[3] J. Pennington, R. Socher, and C.D. Manning. "Glove: global vectors for word representation," in Conference on Empirical Methods on Natural Language Processing 2014, vol. 14. 1532–1543.

[4] P. Bojanowski, E. Grave, and A. Joulin. "Enriching word vectors with subword information," in Transactions of the Association for Computational Linguistics 2016, 5(1).

[5] V. Tshitoyan et al., "Unsupervised word embeddings capture latent knowledge from materials science literature," Springer Nature, 2018, pp. 95–106.

[6] Risch, J. and Krestel, R. (2019), "Domain-specific word embeddings for patent classification," Data Technologies and Applications, Vol. 53 No. 1, pp. 108-122. https://doi.org/10.1108/DTA-01-2019-0002.

[7] Zhang, Y., Chen, Q., Yang, Z. et al. "BioWordVec, improving biomedical word embeddings with subword information and MeSH," Sci Data 6, 52 (2019). https://doi.org/10.1038/s41597-019-0055-0.

[8] I. Chalkidis and D. Kampas, "Deep learning in law: early adaptation and legal word embeddings trained on large corpora," Artificial Intelligence and Law ,vol. 27, pages171–198(2019).