Journal of Data Analytics and Artificial Intelligence Applications, D3AI 1, 1 (January 2025): 84–109

https://doi.org/10.26650/d3ai.003

Submitted: 06.12.2024
Revision Requested: 20.12.2024
Last Revision Received: 02.01.2025
Accepted: 03.01.2025
Published Online 20.01.2025

Journal of Data Analytics and Artificial Intelligence Applications

Research Article Open Access

Analysis of Word Similarities in Tax Laws Using the Word2Vec Method

Ali İhsan Özgür Çilingir[1]

[1] Non-affiliated, İstanbul, Türkiye

Abstract This paper describes word similarity analysis in tax law using the Word2Vec model. By similarity analysis, we mean identifying relationships between similar terms in tax terminology. The Word2Vec model represents the meanings of words with vectors and identifies the semantic relationships of words through the proximity between these vectors.

This article analyzes the semantic proximity of terms frequently used in tax law and visualises the relationships between these words. For example, the close relationships of the word 'mükellef' with words such as 'kişi', 'tam', 'dar', 'firma', and 'imalatçı' are represented through vectors. The paper also explains the mathematical structure of the models. Then, the features of the NumPy, Gensim, Scikit-learn, and Matplotlib libraries of the Python programming language are explained and used for this paper. For the visualisation of the similarity analysis, the t-SNE algorithm, which allows the visualisation of high-dimensional data on a two-dimensional plane, was used.

The main purpose of this paper is to enable AI systems that can be used as tax advisors to better understand tax law by modelling the conceptual relationships between the terms of tax law, thus contributing to the provision of more accurate and consistent information by AI.

Keywords Word2Vec · tax law · natural language processing (NLP) · t-SNE algorithm · Skip-Gram Model · language model visualisation.

Citation: Ali İhsan Özgür Çilingir. 2025. Analysis of Word Similarities in Tax Laws Using the Word2Vec Method. *Journal of Data Analytics and Artificial Intelligence Applications* 1, 1 (January 2025), 84-109. https://doi.org/10.26650/d3ai.003

Corresponding author: Ali İhsan Özgür Çilingir ihsancilingir@gmail.com

1. INTRODUCTION

Tax laws are complex legal texts that regulate the economic structure of a country and the financial obligations of society. These laws contain various concepts and terms that are regularly updated and adapted to the economic conditions. Not only economic and legal experts but also many professionals in different sectors have to understand and apply these laws. However, due to the dense language of tax laws and the abundance of technical terms, these texts are very difficult to understand and analyse. At this point, word similarity analysis using artificial intelligence techniques is a very important tool for making complex terms more understandable and revealing the relationships between laws.

Word similarity analysis has an important place in the field of natural language processing (NLP). Using models such as Word2Vec, these analyses reveal the relationships between similar concepts by representing words as mathematical vectors. Especially in complex and comprehensive texts, similarity analysis makes it possible to determine how related or close terms are. Word similarity analysis in tax laws can serve as a basis for artificial intelligence research in both law and finance, especially in areas such as concept somatisation, automatic classification, and intertextuality.

An in-depth examination of the relationships between word similarity analysis and tax laws will help to better understand legal regulations. Such AI-supported studies make it possible to create a common understanding between different texts, especially by determining the similarity levels of terms that frequently appear across legal texts. For example, inferences such as how specific terms used in tax laws correspond to terms in other legal texts or which concepts are more related to other concepts can also contribute to the economic interpretation of legal regulations.

The purpose of this study is to identify the relationships and similarities between terms used in tax laws and to provide a broader understanding of the meaning of these terms. Since tax laws contain a strict structure and specific linguistic features, analysing these structures can be considered one of the first steps towards the development of AI-supported solutions. The identification of terms used in the same or a similar sense as a result of this analysis can serve as a guide in the interpretation of tax laws and potentially provide a foundation for user-friendly applications.

To achieve this, a corpus of Turkish tax laws was compiled from publicly available legal repositories and subjected to pre-processing steps, including punctuation removal, word form normalisation, and tokenization. The final dataset consisted of 65,258 tokens, providing a balanced representation of key legal concepts. The Word2Vec model was then trained using the Skip-Gram algorithm, with the vector dimensionality set to 100, the context window size set to 5, and the minimum word frequency threshold set to 5. Dimensionality reduction via t-SNE was applied to visualise semantic relationships by projecting high-dimensional embeddings into a two-dimensional space while preserving both local and global data structures.

In addition to improving the comprehensibility of tax laws, this study provides a valuable example of how word similarity analysis can be used in artificial intelligence and law. In the future, these analyses could lead to innovative solutions such as categorising tax legislation in digital environments, automatically highlighting relevant topics, or enabling users to find the information they are looking for faster. Moreover, AI applications developed through such analytics will provide a basis for the creation of new tools that can guide professionals in the interpretation, understanding, and application of tax laws.

2. LITERATURE REVIEW

In their 2013 paper "Efficient Estimation of Word Representations in Vector Space," Google researchers Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean introduced two new model architectures for computing the continuous vector representations of words (CBOW and Skip-Gram Models). While the CBOW model estimates the target word by averaging over the surrounding words, the Skip-Gram model takes a word as input and attempts to estimate the words near it. For example, the models captured semantic similarities with vector operations such as "king - man + woman = queen." Their work also proved that high-dimensional word vectors trained on large datasets such as Google News perform better on many natural language processing tasks [1]. After the publication of the article, many applications were developed, and both local and foreign literature was created on it.

When I decided to write an article on the subject, I first reviewed and benefited from the literature created by IT academics in our country. I can briefly summarise the scope of the studies I benefited from as follows:

In the Master's Thesis titled "Semantic Inference from Turkish Texts Using Deep Learning Approaches" written by Nergis Pervan, the Word2Vec method was used to train the phrases in user comments on social media and e-commerce sites, and the semantic relations of the words in the comments were determined [2].

In the article titled "Turkish Sentiment Analysis Based on Convolutional Neural Network Architectures" written by Aytuğ Onan, sentiment analysis was performed on Turkish texts, and Convolutional Neural Network (CNN) architectures were used. In the article, Word2Vec, FastText, GloVe, and LDA2Vec were used as word embedding techniques, and it was stated that Word2Vec (Skip-Gram model) achieved the highest performance [3].

Murat Tezgider, Beytullah Yıldız, and Galip Aydın's article titled "Improving Word Representation by Tuning Word2Vec Parameters with a Deep Learning Model" aims to improve the classification performance of Turkish texts by tuning Word2Vec parameters with deep learning methods. In this study, different values for parameters such as minimum word count, vector size, and window size were tested for the Word2Vec model. It was observed that the correct choice of these parameters improves the quality of word representation and, therefore, classification success [4].

In the article titled "Similar Sentence Detection Using the Word Embedding Method" written by Mehmet Ali Arabacı, Ersin Esen, Muhammed Selim Atar, Eyüp Yılmaz, and Batuhan Kaltalıoğlu, the Word2Vec model and Fisher coding were combined to detect semantically similar sentences. The method is based on the vectorial representations of the words in the sentence using the Word2Vec model. Then, Fisher coding is applied to create sentence-level vectors. The authors present an effective method that combines Word2Vec and Fisher coding to detect sentence similarity in the Turkish language [5].

Murat Aydoğan and Ali Karcı's article titled "Analysing Word Similarities with Word Representation Methods" aims to identify word similarities in Turkish texts by examining word representation methods. A large Turkish dataset was created, and word relations were analysed using word vector models such as Word2Vec and GloVe. In this study, the CBOW and Skip-Gram algorithms of the Word2Vec method were compared with those of the GloVe method. The Word2Vec method was found to successfully identify the proximity of words and perform better than the GloVe method [6].

In the research article titled "Classification of Turkish News Texts Using Convolutional Neural Networks and Word2Vec" written by Çiğdem İnan Acı and Adem Çırak, the authors showed that Turkish news texts can be successfully classified with Convolutional Neural Networks (CNNs) and Word2Vec and emphasised that these methods make an important contribution to Turkish natural language processing studies [7].

I can briefly summarise the studies of foreign informatics academics from which I have benefited as follows:

Lu XiaoID, Qiaoxing Li, Qian Ma, Jiasheng Shen, Yong Yang, and Danyang Li, in the paper titled "Text classification algorithm of tourist attractions subcategories with modified TFIDF and Word2Vec," investigate an improved text representation method combining TF-IDF and Word2Vec methods and its integration with different classifiers. The aim of this paper is to develop a multi-class classification algorithm by subcategorising tourist attraction description texts and to present a model that provides higher accuracy and stability compared to traditional methods. The authors integrated Word2Vec word embedding methods, TF-IDF (Term Frequency-Inverse Document Frequency), and CRF-POS (Conditional Random Fields) weighting. They collected the descriptions of national A-level tourist attractions in China using web crawler technology and trained the Word2Vec model after pre-processing stages such as word segmentation, grammar tagging, and stop word filtering. Their preferred method was Skip-Gram. Word2Vec is used in this study as a powerful tool in terms of both data representation and classification performance, and they also integrated it with the improved TF-IDF method [8].

Ghislain Wabo Tatchum, Armel Jacques Nzekon Nzeko, Fritz Sosso Makembe, and Xaviera Youh Djam, in their paper titled "Class-Oriented Text Vectorisation for Text Classification: Case Study of Job Offer Classification," discuss class-oriented vectorisation approaches in text classification processes and examine how these methods are more effective in classifying job advertisements. In this paper, preprocessing steps such as data cleaning, tokenization, and stem extraction were performed on job postings. Redundant words or do not carry meaningful information were removed. After these processes, they performed vectorisation using different methods. The vectorisation techniques compared in the paper with traditional methods are TF-IDF, Word2Vec, and Doc2Vec. Class-oriented vectorisation strategies include OC (Occurrence Count), ZIPF, and OWDC (Occurrences Weighted by Dispersion in the Class). In this paper, machine learning models (Naive Bayes (NB), Decision Trees (DT), Support Vector Machines (SVM), and Transformer-based deep neural networks (TFM)) were tested with vectorisation methods. The Word2Vec method was used to represent the text data. One of the prominent results of the paper is that the OWDC strategy generally outperformed the other methods. Furthermore, OWDC provides the highest accuracy rates when used in combination with the TFM (Transformer) model [9].

In the article "Discovery of New Words in Tax-related Fields Based on Word Vector Representation" by Wei Wei, Wei Liu, Beibei Zhang, Rafał Scherer, and Robertas Damasevicius, the authors focus on the detection of new words in tax-related financial texts. Based on the Word2Vec model, the similarity measure of word vectors is used to calculate the similarity in meaning between words. According to the results of the study, this method can be used effectively in large-scale datasets, allowing new words to be automatically added to the dictionary. This method, especially for the discovery of tax-specific terms, has been found to improve the performance of traditional word segmentation tools, contributing to the identification of new words with low frequency but rich in meaning [10].

The article "Deep Learning in Law: Early Adaptation and Legal Word Embeddings Trained on Large Corpora" by Ilias Chalkidis and Dimitrios Kampas examines the early adaptations of deep learning in the legal domain, with a particular focus on the generation of phrases from legal texts. The authors examine the applicability of deep learning in areas such as legal text classification, information extraction, and information retrieval, and emphasise the importance of legal word embedding techniques. This paper describes the impact of word representation in the legal domain with phrases trained on a large legal dataset using the Word2Vec model. This paper provides information about Word2Vec's two main algorithms, Skip-Gram and Continuous Bag of Words (CBOW). The article also highlights how training the Word2Vec model on domain-specific datasets, such as law, improves the model's performance and the accurate capture of semantic relationships between words [11].

In the paper titled "Similarity Analysis of Law Documents Based on Word2Vec" by Chunyu Xia, Tieke He, Wenlong Li, Zemin Qin, and Zhipeng Zou, the authors discuss the use of the Word2Vec model for similarity analysis of legal documents. The different lengths and formats of legal documents create difficulties in similarity analysis. In this context, the authors aimed to perform a more effective similarity analysis by training the Word2Vec model with a specialised dataset of legal documents. Word2Vec learns semantic similarities between words by representing them in a vector space. In the paper, Word2Vec is used to better capture the depth of meaning of words in legal documents. By creating vector representations of sentences and documents, this model allows for more accurate similarity measurements. Skip-Gram tries to predict other words in the context based on a word in the centre. It is especially used to provide more accurate information about rare words. CBOW predicts the centre word based on words in the context and produces more accurate results for more common words. By training the Word2Vec model on legal documents, the authors achieved a 20% higher accuracy than the Bag of Words (BOW) model. It was also observed that the Word2Vec model trained with a dataset specific to legal documents improved the accuracy by 5-10% compared to the model trained with a general dataset. Experiments using methods such as Cosine Similarity and Word Mover’s Distance (WMD) have demonstrated the effectiveness of Word2Vec-based similarity analysis for legal documents [8].

In the paper "Unsupervised Approaches for Measuring Textual Similarity Between Legal Court Case Reports" by Arpan Mandal, Kripabandhu Ghosh, Saptarshi Ghosh, and Sekhar Mandal, the authors examine the use of unsupervised methods for measuring similarity between court decisions. Focusing on the effectiveness of text-based methods, this paper explores how natural language processing techniques such as Word2Vec, Skip-Gram, and CBOW can be used for legal documents. In addition to Word2Vec, the authors also used different methods such as Doc2Vec, TF-IDF, LDA, BERT, Law2Vec, and PScoreVect to measure the similarity between court decisions. Some of these methods (e.g., Doc2Vec and Law2Vec) are direct extensions or adaptations of Word2Vec. While other methods (e.g., BERT, LDA, TF-IDF) aim to achieve the same goal as Word2Vec—representing texts numerically and measuring similarity—they exploit Word2Vec's ability to learn semantic relatedness in different ways. According to the authors, Word2Vec is powerful in capturing semantic similarity between words compared to other methods, but for complex sentence structures or contextual details, more advanced models (e.g., BERT) may be preferable [12].

In the paper "Influence of Various Text Embeddings on Clustering Performance in NLP" by Rohan Saha, the author investigates the impact of different text embeddings on clustering performance in natural language processing (NLP). This study compares the performance of different clustering algorithms with text embeddings such as Word2Vec and BERT using Amazon product review data. The main objective was to evaluate the impact of each embedding method and clustering algorithm on a specific task. The paper emphasises that Word2Vec is a model for learning semantic relations between words. Word2Vec's average vector values (average embeddings) were used. The fact that this method does not include contextual information caused limited performance in some tasks. According to the results of the study, contextual BERT embeddings performed better than Word2Vec in general. However, the performance of the methods differed according to the clustering algorithm [13].

The following information is given in the "Unsupervised Learning (Summer '18)" course note from Columbia University, taught by Ziyuan Zhong and Nakul Verma and authored by Vincent Liu: "t-distributed Stochastic Neighbour Embedding (t-SNE) is a dimensionality reduction technique for visualising high-dimensional data in two- or three-dimensional space. It was developed by Laurens van der Maaten and Geoffrey Hinton in 2008. t-SNE is used in natural language processing to visualise semantic relationships between words by mapping word vectors in low-dimensional space" [14]. The paper also explains the mathematical structure of this algorithm.

In the article "Clustering With T-SNE, Provably" by George C. Linderman and Stefan Steinerberger, t-SNE is described as an optimisation method that minimizes the Kullback-Leibler divergence to cluster high-dimensional data in low-dimensional areas [15].

"An Analysis of the t-SNE Algorithm for Data Visualisation" by Sanjeev Arora and Wei Hu, presented at the Conference on Learning Theory (COLT) 2018, analyzes the use of the t-SNE algorithm for data visualisation. Used to reduce the high-dimensional data to two dimensions, t-SNE visualises and clarifies clusterable data. This paper proves that t-SNE is particularly effective on clusterable datasets with well-separated and global data. The presentation provides the rationale for this success and shows how t-SNE achieves provable success. The authors explain that t-SNE tries to achieve clustering by minimizing the Kullback-Leibler divergence between the similarity vectors of the high-dimensional data and the two-dimensional embedding. This divergence is an optimisation problem that finds the low-dimensional structure that will enable clustering [16].

The article "Visualising Data Using t-SNE" by Laurens van der Maaten and Geoffrey Hinton discusses the t-SNE (t-Distributed Stochastic Neighbour Embedding) algorithm in detail. This paper also describes the Kullback-Leibler Divergence Minimization algorithm. This algorithm compares the similarities between high- and low-dimensional representations with the Kullback-Leibler divergence and minimizes this divergence to keep similar points close and dissimilar points far apart [17].

As mentioned at the beginning, the methods, formulas, and algorithms analysed by the authors mentioned above are used in this paper for our purposes, and we aim to contribute to the literature.

3. DIFFERENCES AND CONTRIBUTIONS OF THE ARTICLE FROM THE REVIEWED LITERATURE

The different aspects of the article compared to the reviewed literature can be summarised as follows:

First, the focus area and application area of the article are different. Most of the studies in the literature have addressed the application of techniques such as Word2Vec in general language processing or other areas (e.g., e-commerce, social media, sentiment analysis). However, this study focuses on a specific legal context, namely tax laws. Although Chunyu Xia et al. conducted similarity analyses for legal documents, this study has chosen a more specific area by focusing specifically on Turkish Tax Laws.

The articles and studies in the literature used general texts or social media data. In this paper, a dataset derived directly from Turkish Tax Laws (‘kanunlarv2.txt’) was used, and a corpus was created for this dataset.

In the literature, the Word2Vec model has been used with different methods (CBOW, Skip-Gram), and in some studies, more advanced models such as BERT have been tried. In this paper, the Skip-Gram method is specifically adapted to analyse semantic relations in tax laws.

Some studies in the literature aim at more general purposes (e.g., analysing user behaviour, discovering new words). This study provides a starting point for a practical application, such as AI-assisted tax counselling. In fact, although Word2Vec was used in this article for the classification of tax law concepts, it can also be used in an AI-assisted consulting (ChatBot) tool to handle concepts from the same class or ensure compatibility between questions and answers. We plan to focus on this in future studies.

In addition to the differences, the contributions of the article to the literature can be summarised as follows:

This article analyzes the conceptual relationships in legal texts by creating a Word2Vec model specific to Turkish Tax Laws. This is a contribution to the studies conducted in the literature on legal documents.

This study provides an infrastructure for developing more effective artificial intelligence-based tax advisory systems using the Word2Vec model. This offers an innovative perspective for both financial and legal applications.

The study has made a significant contribution to the lack of literature on Word2Vec applications for the Turkish language and provides an example of how word vectors suitable for Turkish texts can be developed.

This study proposes a methodology for Turkish natural language processing studies, especially the original word cleaning and simplification processes performed during the corpus generation stages.

Word2Vec and t-SNE are standard techniques widely used in natural language processing and dimensionality reduction. However, the use of a dataset specific to Turkish Tax Laws, the application of these techniques in a legal context, and the focus on practical outcomes such as tax consulting show that this paper makes original contributions to the literature.

This paper is a first step not only in the application of these techniques but also in the integration of more advanced models (e.g., BERT, GPT) to capture the depth of meaning in legal texts. I plan to expand on this by including illegal texts in the analysis in further studies.

4. CHALLENGES AND CONSTRAINTS

During the research and writing of the paper, some unique challenges were encountered when applying the Word2Vec and t-SNE techniques to Turkish Tax Laws. These challenges are outlined below.

Tax laws contain technical language, long sentences, and a dense context. This poses the following unique challenges in natural language processing (NLP) applications. For example, in tax laws, terms such as “mükellef (taxpayer),” “ödeme (payment),” and “muafiyet (exemption)” may have different meanings depending on their context. This increases the risk of semantic inaccuracies when creating a vectorial representation of these terms. Legal texts often contain long sentences and nested structures, making it difficult for the model to learn contextual meanings during corpus creation and the training of word vectors.

The specific difficulties of Turkish are also a significant challenge. Since Turkish is an agglutinative language, the root and affix relations of words pose a particular challenge for models like Word2Vec. The agglutinative structure in Turkish causes words such as “mükellef,” “mükellefiyet,” and “mükellefin” to be represented in different forms. This can result in the meanings of words with the same root being represented by different vectors. To overcome this challenge, a special process was developed to identify word roots and remove unnecessary suffixes during the corpus creation process.

Turkish characters and encoding issues should also be considered a challenge. Turkish characters such as "Ğ," "İ," "Ş," and "Ü" can create technical problems when reading and processing the dataset. The paper suggests that different character encodings (e.g., UTF-8, ISO-8859-9) should be tried to resolve these problems.

In legal texts, there are contextual relationships between concepts that are not explicitly stated. For example, there are indirect concept relationships. Legal terms are often related to concepts that are not explicitly stated but are linked in meaning. For instance, words such as "vergi (tax)," "beyanname (declaration)," and "tahakkuk (accrual)" are closely related in legal processes, but this relationship is implicit in the text. Modelling such conceptual links is a challenge that exceeds Word2Vec's structure based on direct word relationships.

The same term can mean different things in different laws or contexts. For instance, the word "vergi" may be associated with "beyanname" in one context and with the concept of "ceza (penalty)" in another. Modelling these contextual differences makes the training process of the model more complex.

The paper worked with a specific dataset, such as the Turkish Tax Laws. However, the problems that arose during the organisation and processing of legal texts posed a unique challenge. Tax laws often contain fragmented information contained in different documents. Combining them into a single dataset and creating a coherent corpus is a time-consuming and laborious process. The paper solved this problem by bringing all the laws together in a file named "Kanunlarv2.txt."

When creating a corpus, conjunctions, pause marks, and word fragments that do not make sense need to be weeded out. This process required both technical and linguistic expertise. This study uses special patterns and filtering methods to extract such words.

The t-SNE algorithm may lose some relationships in the high-dimensional data. For instance, multiple contextual relationships between two terms may not be fully reflected in the low-dimensional plane. However, the model results were satisfactory. Indeed, in the visualisation obtained by applying the t-SNE algorithm, the proximity between the word "mükellef" and words such as "şirket (company)" and "dar (narrow)" can be clearly seen. However, what these relationships mean in a legal context requires legal knowledge and interpretation beyond visualisation. In future studies, the results of models such as Law2Vec, BERT customised for law, Doc2Vec, and GPT can be compared with Word2Vec to explore this issue in more depth.

The dataset of Turkish Tax Laws consists of 65,258 words. It is known in the literature that the Word2Vec model establishes stronger semantic relations when trained with much larger datasets. However, since 65,258 words is considered sufficient for general natural language processing projects, we did not see any harm in building the paper on this. This paper is a starting point in its field, and future work could include not only the law but also the broader tax literature.

The Word2Vec model prioritises frequently used words in the training data. Therefore, frequent terms such as "vergi," "beyanname," and "ödeme" may have a stronger representation in the model than other rare terms. This may lead to rare terms or more complex contexts being ignored. As explained in the research methodology section, we took this constraint into account and removed pause words and reduced some words to their roots to avoid omission due to Turkish suffixes.

5. MATHEMATICAL STRUCTURE OF THE WORD2VEC MODEL

5.1. General Information

The Word2Vec model represents words in vector space by using statistical relationships while building word vectors. The main purpose of creating word vectors is to represent each word with a vector of fixed size. These vectors are used to detect the relationships of a word with other words. Initially, each word is represented by an arbitrary vector. For example, each word in the dictionary is assigned a vector of a certain size. The model learns using the "distributional hypothesis," which assumes that similar words appear in similar contexts. During the training process, these vectors are optimised, and the semantic relationships between words are reflected in the optimised vectors. The model learns which words should appear in different contexts and assimilates the internal structure of the language. Both versions are suitable for most applications. Mueller and Massaron stated that the Skip-Gram version is better at representing rare words [18].

The Word2Vec model, developed by Tomas Mikolov and his team, includes two separate models: CBOW and Skip-Gram. "The Continuous Bag of Words (CBOW) model learns word vectors in the projection layer and predicts the central word using words in the context. This architecture predicts the central word based on other words in the context. The input layer creates a projection of the surrounding words and uses a weighted output layer to predict the central word based on this projection. In the "Continuous Skip-Gram Model," for each word, the surrounding words are predicted [1]. In this paper, Tomas Mikolov and his team aim to produce high-quality word vectors that best represent the semantic and syntactic similarities of words. The model converts related words into vectors through mathematical processing and thus detects the similarity between, for example, "king" and "queen" and, as we exemplify in this paper, “mükellef” ile "kişi," "tam,” “dar,” “firma,” “imalatçı,” etc.

Representations (words) whose semantic proximity is made through word vectors (word embeddings) are used as neural network inputs. This makes it possible to express the meanings of language in numerical form . As a result of this process, words close in meaning appear as similar vectors [19], [20].

5.2. CBOW model

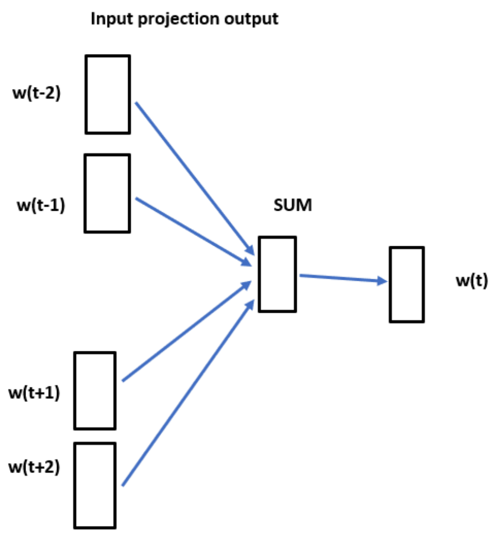

Figure 1. CBOW Model Image

The meaning and mathematical model of this notation can be summarised as follows: Figure 1 provides a simple visualisation of the working principle of the CBOW model. In the CBOW model, for example, the target word w(t) and the words in the context (e.g. w(t-2), w(t-1), w(t+1), w(t+2)) are taken as input. The vectors of these words are summed in a projection layer to form an average vector. This average vector is then used to predict the target word w(t) at the output.

5.2.1. CBOW Stages

5.2.1.1. Collection of Contextual Words

$$v_{\text{context}} = \frac{1}{2m} \sum_{j=-m}^{m} v_{w+j}, \; j \neq 0 \tag{1}$$

- v_{context} : Context vector. It represents the environment (context) in which the word appears. This vector is calculated based on the other words in the context.
- $\frac{1}{2m}$: Normalisation factor. 2 m represents the total number of words in the context window, consisting of m words to the left and m words to the right. When averaging, the total vectors are divided by this number.
- $\sum_{j=-m}^{m}$: refers to the summation. It sums the vectors of all words to the left (-m) and to the right (+m), excluding the center word itself ($j = 0$) or when ($j \neq 0$).
- v_{w+j}: the vector of a context word at position j (either to the right or left) of w. For example, j = −1 represents the vector of the word preceding w, and j = +1 represents the vector of the word following w.
- $j \neq 0$: This condition ensures that the centre word itself ($j = 0$) is excluded from the summation. In other words, only the surrounding words contribute to the context vector.

5.2.1.2. Predicting the Target Word

The context vector ($_{\text{vcontext}}$) calculated in the first stage is used to predict the target word. This calculation is done with the Softmax function[1].

$$\mathbf{P(w_{\mathit{t}}/\ context)} = \frac{\mathbf{exp(v_{context}^{T}\ v_{wt})}}{\mathbf{\sum_{w \in W} exp(v_{context}^{T}\ v_{w})}} \tag{2}$$

In this formula:

- $P(w_t/\ \text{context})$: Represents the probability that the word w_t occurs in a given context (context). This measures the proximity of the context to the word w_t .
- v_{context} : The context vector. It is a vector obtained by combining the vector representations of the words that constitute the context.
- w_{wt} : The vector representation of the target word (w_t). This vector represents the meaning or features of the word in numerical form.
- $v_{\text{context}}^{T} v_{\text{wt}}$: The dot product of the context vector and the target word vector.
- $T : v_{\text{context}}^{T}$: refers to the transpose of the context vector.
- $\exp(v_{\text{context}} v_{\text{wt}})$ The exponential function (exp) is applied to the dot product. The exponential function amplifies the effect of large values and reduces the effect of small and negative values. Thus, the “relationship” between the context and the word becomes more pronounced.

[1]The Softmax activation function is a generalisation of logistic regression that can be applied to continuous data instead of binary classification. It is often included in the output layer of a classifier because it produces output for more than two classes [36]

- W : Vocabulary. It is the set of all words on which the model operates.
- $\sum_{w\in W}\exp(v_{\text{context}}^{T} v_w)$ This denominator expression represents the sum of the exponential values for all words in the vocabulary (W) in relation to the context. This is the normalisation step of the softmax function. The exponential value of each word associated with the context is summed, and the result is normalised so that the total probability of all words' association with the context equals 1.

5.2.1.3. Updating Vectors

The loss function is minimized to increase the probability of correctly predicting the target word. For example, a negative logarithmic loss function is used:

$$J = \log P(w_t / \text{ context}) \tag{3}$$

J: The symbol for the loss function. It measures the performance of the model. The goal of the CBOW model is to match the target word (w_t) with the context by minimizing this loss. The lower the value of J, the more successful the model is in predicting the correct word from the context.

P (w_t / context) : The probability that the target word (w_t) will occur given the context. A low value of P (w_t / context) results in a large loss (J), , while a high value of P (w_t / context) results in a small loss, indicating that the model is making better predictions.

This loss function measures the relevance of the target word to the context in the CBOW model and is used to optimise the relationship between the context and the target word. Through backpropagation, the target and context vectors are updated to minimize J. This process enables the model to learn relationships within the language.

5.3. Skip-Gram Model

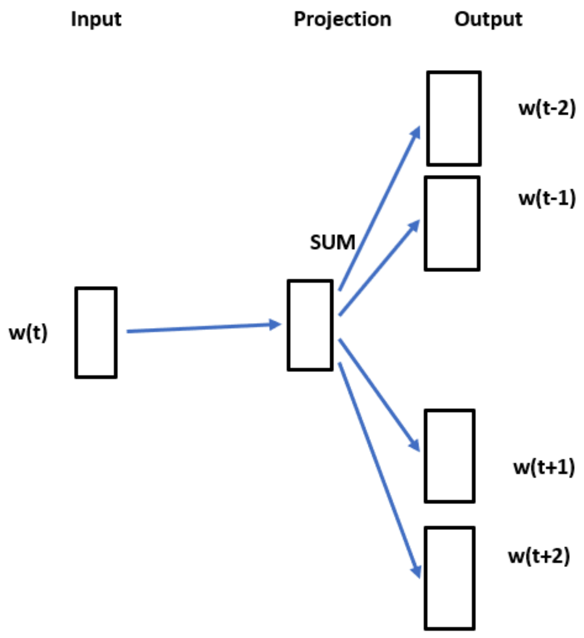

Figure 2. Skip-Gam Model Image

The Skip-Gram model represents words as vectors and estimates the proximity of one word to another using the dot product of their vectors. The CBOW model predicts a word based on its surrounding words. These two algorithms together create a model that represents each word as a vector. The mathematical foundation of these vectors is that the proximity of words in the vector space reflects their semantic similarity [19].

The meaning and mathematical model of this notation can be summarised as follows:

Figure 2 illustrates the working principle of the Skip-Gram model. In this model, the centre word w(t) is taken as input, and based on this word, the surrounding words w(t-2), w(t-1), w(t+1), and w(t+2) within a given context are predicted. The main purpose of the Skip-Gram model is to predict the surrounding context words from the given centre word. This model performs particularly well with large datasets and can also produce good results for rare words. It establishes matches between target words and context words: the target words are the input, and the context words are the output.

The Skip-Gram model is a shallow neural network consisting of an input layer, an embedding layer, and an output layer. The goal of the model is to generate an output probability distribution vector given a target word input. This probability distribution vector (which sums to 1) reflects the likelihood of each word appearing in the context window of the target word. The probability is high for words that share the same context and low for words that do not. Once trained, the model only requires its weights.

To obtain useful vector embeddings, the initially random weights in the model need to be optimised. This optimisation process is carried out to minimize the loss function.

The loss function and its description are provided below :

$$J = \sum_{t=1}^{T} \sum_{-m \le j \le m} \log\left(P\left(w_{t+j}/w_t\right)\right) \tag{4}$$

J: The Loss Function.

T: The length of the text.

m: The window size.

$P\left(w_{t+j}/w_t\right)$: The probability of obtaining the context word given the target word.

This equation represents a nested loop, where you iterate through all (target word, context word) pairs and sum their probabilities. The minus sign is used as part of the machine learning process to minimize the value of the loss function.

Calculation of probabilities: To calculate the probability distribution, the Softmax function is used, which considers the dot product of the target embedding vector and the embedding vectors of each word in the vocabulary.

The dot product of the two vectors can be expressed as

$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$

The function that provides the probability distribution can be expressed as follows:

$$p(\text{context word/ target word}) = exp\left(u_{\text{target}}^T v_{\text{context}}\right) / \sum_{w=1}^{\text{Word}} \left(\exp\left(u_{\text{target}}^T v_{\text{context}}\right)\right) \tag{5}$$

In this formula

p(context word/target word): Represents the probability of observing a "context word" given a "target word." This means that the model attempts to predict which words are likely to appear around a target word.

$\mathrm{e\ xp}(\ \mathbf{u}_{\mathbf{target}}^{\mathbf{T}}\ \mathbf{v}_{\mathbf{context}})$ **:** Here, an inner product is computed, representing the relationship between the target word and the context word. u_{utarget} and v_{context} are vectors of a given size for the target and context words. The inner product of these vectors is calculated, and the result is passed through an exponential function. The exponential function amplifies the closeness between words in the model, assigning higher probabilities to closer words.

Division Operation (/): This quotient is used to normalise the probability, ensuring that the total probability of all words is equal to 1. In the denominator, a similar calculation is performed for all words (W) in the vocabulary, and the result is used to normalise the probability.

Total $\left(\sum_{w=1}^{Word}\left(exp\left(u_{target}^{T}\right)v_{context}\right)\right)$ This summation normalises the calculation for every word in the vocabulary. It provides a probability distribution of the context word given the target word across all possible context words.

6. AUXILIARY ALGORITHMS

6.1. t-SNE Algorithm

t-SNE (t-distributed Stochastic Neighbour Embedding) is a dimensionality reduction technique used to reduce high-dimensional data to a low-dimensional space, particularly for visualising complex structures in datasets. It was developed in 2008 by Laurens van der Maaten and Geoffrey Hinton. The primary purpose of t-SNE is to project data into a lower-dimensional space (usually 2 or 3 dimensions) while preserving the similarities in the high-dimensional data. This transformation allows the data to be represented as graphs or visuals that are easier for humans to interpret [17].

The mathematical steps of the algorithm are described as follows [14], [15], [16]:

In the first step, the similarities between two data points in high-dimensional space are calculated. For each data point xi and xj in the high-dimensional space, the probability that xj is a neighbour of xi is calculated using a Gaussian distribution (normal distribution). The probability is defined as follows:

$$Pj/i = exp(-\left\|xi - xj\right\|^2)/2\sigma\_1^2)/\sum k \neq iexp(-\left\|xi - xk\right\|^2/\sigma\_1^2)] \tag{6}$$

In this formula:

- ‖xi-xj‖² : Square of the Euclidean distance between xi and xj.
- σi : The bandwidth parameter is selected depending on the data point xi.
- pj|i : The probability that xj is a neighbour of xi.

These probabilities are symmetrised for each data point as follows:

Pij = (pj/i + pi/j) / 2N

Where N is the total number of data points in the dataset. This symmetric form ensures that the relationships between the two data points are equalised.

Similarity in Low-Dimensional Space (with t-distribution): The t-distribution is used to transfer these similarities from high-dimensional space to low-dimensional space. The similarity between the two low-dimensional point yi and yj is calculated as follows:

$qij = (1+ ||yi - yj||^2)^{-1}) / (\Sigma k \neq i (1+ ||yk - yj||^2)^{-1})$

In this formula:

‖yi-yj‖²: Square of the Euclidean distance between yi and yj.

qij: The probability of similarity between yi and yj in low-dimensional space.

Because the t-distribution has wider tails, it better distinguishes distances between distant points and more effectively reflects the structure between clusters.

6.2. Kullback-Leibler Divergence Algorithm

The Kullback-Leibler Divergence (KL Divergence) is an information-theoretic metric used to measure the difference between two probability distributions. Specifically, it helps us understand how "far" one probability distribution is from another. KL Divergence typically measures the difference in information between a reference distribution (true distribution) and a predicted distribution (approximation distribution).

The Mathematical Formula for the Kullback-Leibler Divergence:

Let P(x) and Q(x) be two probability distributions. The KL divergence is defined as follows:

$$DKL(P \parallel Q) = \sum\_X[P(x)\log(P(x)/Q(x))] \tag{7}$$

In this formula:

P(x) : The true distribution or reference distribution (e.g. distribution derived from data).

Q(x) : The approximation distribution or model distribution,

DKL(P‖Q) : KL Divergence result.

KL Divergence is used in machine learning to measure the difference between a model's predicted distribution and the actual distribution. In the field of natural language processing, it is employed to evaluate how well the estimated distributions of language models align with the actual data (Bissiri & Walker, 2012, pp. 1139-1160).

7. PYTHON LIBRARIES WE USE

NumPy:

NumPy is a fundamental package for scientific computing in Python. It provides a multidimensional array object, various derived objects (e.g., masked arrays and matrices), and numerous routines for fast operations on arrays, including mathematical, logical, shape processing, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, and random simulation.. In the Word2Vec implementation, NumPy was chosen for its performance optimisation, ease of use, and ability to provide mathematical tools that support natural language processing. It offers a significant speed and efficiency advantage over performing the same operations in pure Python.

Gensim:

Gensim is a Python library for topic modelling, document indexing, and similarity retrieval with large corpora. It is primarily designed for the natural language processing (NLP) and information retrieval (IR) communities (https://pypi.org/project/gensim/, 2024). Gensim is an essential tool for training and implementing the Word2Vec model to quickly and easily generate vectors without going into complex mathematical operations and data preprocessing details. It is simple to use for model training and querying. For example, obtaining the vector of a word or finding similar words is possible with just a few lines of code.

Scikit-learn:

Scikit-learn is an open-source and powerful Python library for machine learning and data analysis. It enables the easy implementation of statistical modelling, data preprocessing, and supervised and unsupervised learning algorithms. While Scikit-learn is not directly used in Word2Vec projects, it can serve as a complementary tool. For example, Scikit-learn's tools such as CountVectorizer or TfidfVectorizer can be used to clean and tokenise text data and convert textual labels (LabelEncoder) into numerical data. Additionally, we used the t-SNE (t-Distributed Stochastic Neighbour Embedding) algorithm from Scikit-learn to visualise the similarities between words.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualisations in Python. The Word2Vec project serves as a powerful tool for visualising word embedding vectors. Since the vectors generated by Word2Vec are often multidimensional, visualisation plays a crucial role in analysing and interpreting these vectors. Matplotlib enables us to represent semantic similarities between words by clustering words with similar meanings in the same graph. Additionally, it supports the implementation of dimension reduction algorithms such as t-SNE and optimising the cost function, Kullback-Leibler Divergence, during the reduction of high-dimensional vectors into 2D or 3D space. These features were the primary reasons for using this library in our research.

8. RESEARCH METHODOLOGY

8.1. Generation of the Tax Law Dataset

The dataset comprises primary and secondary tax law texts. It was retrieved from https://www.gib.gov.tr/gibmevzuat and is stored on our computer hard drive in .txt format under the file name "Kanunlarv2.txt". As of 15.10.2024, the file includes the following laws: Tax Procedure Law (Vergi Usul Kanunu), Income Tax Law (Gelir Vergisi Kanunu), Corporate Tax Law (Kurumlar Vergisi Kanunu), Value Added Tax Law (Katma Değer Vergisi Kanunu), Stamp Duty Law (Damga Vergisi Kanunu), Motor Vehicles Tax Law (Motorlu Taşıtlar Vergisi Kanunu), Law on Collection of Public Receivables (Amme Alacaklarının Tahsili Hakkında Kanun), Expense Tax Law (Gider Vergileri Kanunu), Law on Valuable Papers (Değerli Kağıtlar Kanunu), Law on Real Estate Tax (Emlak Vergisi Kanunu), and the Law on Municipal Revenues (Belediye Gelirleri Kanunu). The dataset consists of 65,258 words and word fragments.

8.2. Downloading Related Python Libraries

As explained above, we downloaded the following Python libraries: NumPy, for scientific computing; Gensim, which includes the Word2Vec formulation; Scikit-learn, which provides the t-SNE algorithm; and Matplotlib, for creating graphs of similar word vectors by reducing their dimensionality.

8.3. Opening and Reading the Kanunlarv2.txt File with Turkish Character Encoding

The Turkish characters in the words of the Turkish Tax Law in the Kanunlarv2.txt file were read using different character encodings, including "utf-8," "ISO-8859-9," "windows-1254," and "ISO-8859-1." The most appropriate encoding was selected, and the file was opened, read, and its contents printed. To accomplish this, a loop was created to try each encoding in turn. Once the correct encoding was identified, the loop was terminated, and the file was opened in the read mode.

A small portion of the output is shown below:

"Kanun: 213 - VERGİ USUL KANUNU Yeni Pencerede Aç Yazdır GİRİŞ Kanunun şümulü Madde 1 Bu kanun hükümleri ikinci maddede yazılı olanlar dışında, genel bütçeye giren vergi, resim ve harçlar ile il özel idareler-

ine ve belediyelere ait vergi, resim ve harçlar hakkında uygulanır. Yukarıda yazılı vergi, resim ve harçlara bağlı olan vergi, resim ve zamlar da bu kanuna tabidir. Bu kanunun hükümleri kaldırılan vergi, resim ve harçlar hakkında da uygulanır. Gümrük ve tekel vergileri Madde 2 (Değişik: 23/1/2008-5728/271 md.) Gümrük idareleri tarafından alınan vergi ve resimler bu Kanuna tabi değildir. Bu vergi ve resimlerle ilgili olarak 27/10/1999 tarihli ve 4458 sayılı Gümrük Kanununun 242 nci maddesi hükümleri uygulanır. Vergi Kanunlarının Uygulanması ve İspat: Madde 3 (Değişik: 30/12/1980 - 2365/1 md.) A) Vergi kanunlarının uygulanması: Bu Kanunda kullanılan "Vergi Kanunu" tabiri işbu Kanun ile bu Kanun hükümlerine tabi vergi, resim ve harç kanunlarını ifade eder. Vergi kanunları lafzı ve ruhu ile hüküm ifade eder. Lafzın açık olmadığı hallerde vergi kanunlarının hükümleri, konuluşundaki maksat, hükümlerin kanunun yapısındaki yeri ve diğer maddelerle olan bağlantısı gözönünde tutularak uygulanır. B) İspat: Vergilendirmede vergiyi doğuran olay ve bu olaya ilişkin muamelelerin gerçek mahiyeti esastır. Vergiyi doğuran olay ve bu olaya ilişkin muamelelerin gerçek mahiyeti yemin hariç her türlü delille ispatlanabilir. Şu kadar ki, vergiyi doğuran olayla ilgisi tabii ve açık bulunmayan şahit ifadesi ispatlama vasıtası olarak kullanılamaz. İktisadi, ticari ve teknik icaplara uymayan veya olayın özelliğine göre normal ve mutad olmayan bir durumun iddia olunması halinde ispat külfeti bunu iddia eden tarafa aittir. BİRİNCİ KİTAP Vergilendirme BİRİNCİ KISIM Genel esaslar BİRİNCİ BÖLÜM Vergi uygulanmasında yetki Vergi dairesi Madde 4 Vergi dairesi mükellefi tesbit eden, vergi tarh eden, tahakkuk ettiren ve tahsil eden dairedir.....”

8.4. Listing the Frequencies (Raw Frequencies) of the Words in the Text

To prepare the study, we listed the word frequencies in the “kanunlarv2.txt” file using Python.

Table 1. Frequency Table of the Top Twenty Words

	Word	Frequency
0	ve	9320
1	sayılı	3963
2	bu	3895
3	ile	3272
4	veya	3208
5	kanunun	3045
6	madde	2526
7	vergi	2368
8	maddesiyle	2181
9	yürürlük	1735
10	için	1639
11	bir	1611
12	vergisi	1341
13	göre	1282
14	önceki	1227
15	olarak	1200
16	değişen	1083
17	kadar	1082
18	1	1020
19	edilen	1013

As previously mentioned, there are a total of 65,258 words and word fragments in the "kanunlarv2.txt" file. In the frequency table above, as shown in Table 1, the first 20 entries account for 48,011 occurrences, and 3,709 of these are variations of the word “vergi,” forming meaningful expressions like “vergisi.” The remaining entries consist of non-conceptual words, conjunctions, or word fragments. Therefore, this text needs to be analysed for semantic relationships using machine learning and deep learning algorithms. It is necessary to filter out word fragments and meaningless words to create a meaningful subject for analysis.

8.5. Create a Corpus (Collection of Meaningful Words)

At this stage, the goal is to open the text file kanunlarv2.txt using Python code, read it with specific character encodings, and clean the text to create a corpus. The stages of the process are as follows:

- The text is split into lines using `text.split('\n')`, and these lines are stored in a list called t_list. Each line is added to the list as a new item.
- An empty list is created with the code corpus = []. This list contains the cleaned words from the processed text.
- A pattern named pattern is created. This pattern includes various punctuation marks (.,!?;:…"“'\"(){}[]-<>|/@#$%^&*_=+~) and digits (1234567890) that may appear in the text. These characters and numbers have been removed from the text.
- A list named Delete_words is created. This list contains words that may be present in the text and need to be deleted. Both uppercase and lowercase variations of these words are included. For example, "birinci," "BİRİNCİ," "İlgili," Roman numerals, and some unwanted characters (e.g., 'x96') are included. Meaningless suffixes such as 'sine, 'ine, aa, 'una are also removed as they could interfere with frequency calculations and distort vector calculations.
- The code snippet `re.sub(pattern, ", cumle)`, is used to remove the punctuation marks and digits from each sentence.
- The snippet `re.sub(r'\b' + kelime + r'\b', ", temizlenmis_cumle, flags=re. IGNORECASE)` is used to remove each word in the Delete_words list from the text in a case-insensitive manner.
- In the corpus, certain words were transformed into root forms or meaningful common words. For example, "kurumları" was replaced with "kurum," and "cezanın" was replaced with "ceza." A total of 2,490 words underwent this process. This process, referred to as meaningful simplification;, is based on over 25 years of expertise in tax law, more than 11 years as a doctor of tax law, and our understanding of corpus creation.
- Using the `split()` code snippet, the cleaned sentences were split into individual words and added to the corpus list.
- Finally, the first 50 items of the corpus (each item being a string of words) were printed on the screen using `print(corpus #r[50])`.

The code was successfully executed, and the output was obtained, as shown in Figure 1.

```
  [['Kanun', 'vergi', 'USUL'], ['Yeni', 'Pencerede', 'Aç'], ['Yazdır'], [], [], ['GİRİŞ'], [],
[], [], ['şümulü'], [], ['kanun', 'bütçeye', 'giren', 'vergi', 'resim', 'harç', 'il', 'özel',
'idarelerine', 'belediyelere', 'vergi', 'resim', 'harç'], [], ['Yukarıda', 'vergi', 'resim',
'harç', 'vergi', 'resim', 'zam', 'kanuna', 'tabidir'], [], ['kaldırılan', 'vergi', 'resim',
'harç'], [], [], ['Gümrük', 'tekel', 'vergi'], [], ['Gümrük', 'idareleri', 'alınan', 'vergi',
'resimler', 'Kanuna', 'değildir', 'vergi', 'resimlerle', 'Gümrük'], [], ['vergi', 'Kanunlarının',
'Uygulanması', 'İspat'], [], ['vergi', 'kanunlarının', 'uygulanması', 'Kanunda', 'kullanılan',
'vergi', 'tabiri', 'işbu', 'Kanun', 'Kanun', 'vergi', 'resim', 'harç', 'kanunlarını', 'ifade'],
[], ['vergi', 'kanunları', 'lafzı', 'ruhu', 'ifade', 'Lafzın', 'açık', 'olmadığı', 'hallerde',
'vergi', 'kanunlarının', 'konuluşundaki', 'maksat', 'hüküm', 'yapısındaki', 'yeri', 'maddelerle',
'bağlantısı', 'gözönünde', 'tutularak'], [], ['İspat', 'vergi', 'vergiyi', 'doğuran', 'olay',
'olaya', 'muamelelerin', 'gerçek', 'mahiyeti', 'esastır'], [], ['Vergiyi', 'doğuran', 'olay',
'olaya', 'muamelelerin', 'gerçek', 'mahiyeti', 'yemin', 'türlü', 'delille', 'ispatlanabilir',
'vergiyi', 'doğuran', 'olayla', 'ilgisi', 'tabii', 'açık', 'bulunmayan', 'şahit', 'ifadesi',
'ispatlama', 'vasıtası', 'kullanılamaz'], [], ['İktisadi', 'ticari', 'teknik', 'icaplara',
'uymayan', 'olayın', 'özelliğine', 'normal', 'mutad', 'durumun', 'iddia', 'olunması', 'ispat',
'külfeti', 'bunu', 'iddia', 'tarafa', 'aittir'], [], [], ['KİTAP'], ['Vergilendirme'], [], [],
['esaslar'], [], [], ['vergi', 'uygulanmasında', 'yetki'], [], [], [], ['vergi', 'dairesi'], [],
['vergi', 'dairesi', 'mükellef', 'tesbit', 'vergi', 'tarh', 'tahakkuk', 'ettiren', 'tahsilat',
'dairedir'], []]
```

Figure 1. A Sample of the Output

8.6. Listing of the Corpus Frequencies

After performing the necessary cleaning and corrections in the text, the frequency report of the top 20 words was generated, as shown in Table 2.

Table 2. Frequency Table of the Top Twenty Words

	Word	***Frequency***
0	vergi	4603
1	değişme	1083
2	gelir	1007
3	ödeme	997
4	kurum	997
5	oran	844
6	değer	748
7	geçici	642
8	mükellef	608
9	hesap	545
10	işletme	529
11	ceza	520
12	mal	498
13	kazanç	493
14	tahsilat	485
15	özel	450
16	sermaye	437

	Word	**Frequency**
17	indirim	423
18	hizmet	422
19	beyan	406

When this list is examined, it is observed that the words and symbols in the "kanunlarv2.txt" file, which are not suitable for meaningful similarity analysis (e.g., "ve", "bu", "veya", "sayılı", "için", "1"), have been eliminated, making the text much more suitable for creating a similarity model.

8.7. Creating Word2Vec (Model)

At this stage, the necessary steps can be taken to create a similarity model:

To build a similarity model, it is essential to train the Word2Vec model on the words in the Tax Law. The goal of the training is to teach the model the meanings of the words in the corpus. For this purpose, 100-dimensional vectors were created using the Skip-gram algorithm, and 5 words[2] were considered within the context of a target word. The model only included words that appeared at least 5 times.[3] Once trained, the model generated vector representations of the words, enabling a better understanding of the relationships and similarities between the words.

We can summarise this explanation in terms of the process stages as follows:

- The corpus consists of cleaned sentences converted into a list of words. The model will learn the relationships between words in this corpus.
- Each word is represented by a 100-dimensional vector. Higher vector sizes allow the model to capture more detailed information but require greater computational power.
- The window size determines how many words around a target word will be used as the context. Here, the window size is set to 5, meaning the model considers up to 5 words to the left and right of a target word for learning.
- A word must appear at least 5 times to be included in the model, while less frequent words are ignored. This prevents rare words from affecting the model.
- It is specified whether the model will use the Skip-gram (sg=1) or CBOW (sg=0) algorithm. In this case, the Skip-gram algorithm is used with the sg=1 option. Skip-gram predicts other words in the context of a target word and performs better, especially on small datasets.

8.8. Vectorial Representation of the Word

This stage involves reporting the vectorial representation of the words. In other words, it is the stage where the trained model's output is retrieved from the selected word vectors trained in the Word2Vec model, allowing us to understand how the model represents the words. The word chosen for this analysis is *'mükellef'*, which is one of the fundamental concepts in tax terminology.

Let us explain this in detail:

[2]Very rare words are often misspellings, conjunctions, or trivial terms. Excluding words with a frequency lower than 5 enables the model to learn in a more meaningful way. Wider windows (e.g., >10) may associate unnecessary words when learning context relationships, while narrower windows (e.g., <3) fail to capture sufficient depth of meaning.

[3]Using too high a frequency threshold (e.g., >10) may exclude underutilised but significant legal terms. For example, even if a legal term like "obligation" is mentioned infrequently, it is crucial for the analysis.

As mentioned at the beginning of this article, the Word2Vec model represents each word as a vector of a certain size. These vectors are trained to capture the semantic and contextual similarities between words. For example, words with similar meanings are represented by vectors that are mathematically close to each other. In this context, the following operations were performed to analyse the similarity of the word *'mükellef'* .

In the Word2Vec model, the area containing the vectors created for words after training is referred to as model.wv in the code. The model.wv object stores the vectors corresponding to the words. For example:

- `model.wv['kelime']` returns the vector for that word.
- `model.wv['mükellef']` returns the vector that the model assigns to the word *'mükellef'*.

This vector is an array of numbers that mathematically represents the meaning of the word *'mükellef'* based on the context the model has learned. Since we set vector_size=100, the word *'mükellef'* is represented by a vector with 100 elements. This vector encodes the contextual relationships of the word *'mükellef'* with other words. For example, *'mükellef'* has vectors similar to words with related meanings, such as *'tam'* and *'dar'*. These vectors can be used to

- Find similar words.
- Measure word similarity.
- Create word clusters.

The output of the word vector appears in Figure 2 (100 items).

```
array([-6.23710275e-01, -4.63944301e-02,  4.93801236e-01,  5.19869626e-01,
        2.64085293e-01, -4.01404612e-02,  4.53982145e-01,  4.04999465e-01,
       -2.52363645e-02, -2.29218736e-01,  2.70977706e-01, -6.21370435e-01,
        4.42850173e-01,  1.88748702e-01,  4.23074663e-01, -2.48641014e-01,
       -3.55076268e-02, -1.82925805e-01,  1.52746215e-01, -3.49068701e-01,
        2.41844654e-02,  3.93145740e-01,  3.52574557e-01,  5.11918101e-04,
        1.96201742e-01,  7.41723031e-02, -2.12855414e-01, -4.91001159e-02,
       -5.81765920e-03, -4.18194771e-01,  8.73388574e-02,  1.11241512e-01,
        2.79387623e-01,  2.63844766e-02,  9.17030573e-02,  5.33403814e-01,
        2.66016543e-01, -1.37306616e-01, -7.47924373e-02, -4.67420578e-01,
       -3.28616440e-01,  1.10134447e-03, -1.01418853e-01, -3.93146545e-01,
        8.64220336e-02,  1.79907549e-02, -4.13307697e-01, -1.48787230e-01,
        7.44897947e-02,  1.75265461e-01,  3.55105400e-02, -2.73291394e-02,
       -1.50529653e-01, -4.85246740e-02,  1.54171020e-01, -1.68570891e-01,
        3.48834060e-02, -9.14018080e-02, -4.10086721e-01, -7.69241899e-02,
       -1.77023351e-01, -1.62242562e-01,  5.14853597e-01, -1.14671946e-01,
       -5.61786294e-01,  5.57065085e-02,  8.95505175e-02,  3.13859880e-01,
       -3.32743734e-01,  3.15872729e-01, -2.60060638e-01,  1.40525192e-01,
       -1.07785322e-01,  1.95944458e-01, -9.53617766e-02, -1.41765758e-01,
       -1.98408827e-01, -2.68165112e-01, -3.73588473e-01,  1.27587229e-01,
       -1.75960913e-01,  1.62031755e-01,  2.75125414e-01,  6.89642057e-02,
       -4.20767553e-02, -1.14048988e-01,  1.91145271e-01,  2.70561635e-01,
        3.33638400e-01,  4.17892754e-01,  2.03026727e-01, -4.98898625e-02,
        2.53186584e-01, -1.00524463e-01,  6.04738295e-01,  4.28849995e-01,
        2.33635247e-01, -8.20747167e-02, -1.21545447e-02,  3.45590383e-01],
      dtype=float32)
```

Figure 2. Representation of the 100-Element Output of the 'mükellef' Vector

8.9. Word Similarity Analysis

At this stage, we retrieved the words that are vectorially most similar to the word *'mükellef'* and their similarity scores. Cosine similarity is commonly used to measure the similarity between words. Using the `most_similar()` function in Python, we calculated the vector similarity between the target word (*'mükellef'*) and other words, returning a ranked list of words with the highest similarity scores.

The result includes the similarity score of each word relative to the word *'mükellef'*. These scores typically range between −1 and 1. A score closer to 1 indicates a high degree of similarity to the word *'mükellef'*, while a score of 0 indicates very little semantic similarity.

This analysis is particularly useful for understanding semantic relationships between words in a language. It can be applied to

- Find words with similar meanings.
- Analyse language models.
- Develop automatic text recommendation systems.

The output of the analysis is shown in Figure 3.

```
[('dar', 0.8771734833717346),
('tam', 0.8355701565742493),
('imalatçı', 0.829497754573822),
('firma', 0.8249997496604919),
('ödenen', 0.8103259801864624),
('kişiler', 0.7855820059776306),
('kişi', 0.7799397706985474),
('firmalara', 0.7753022909164429),
('mükellefiyeti', 0.7662615180015564),
('üreten', 0.7659305930137634)]
```

Figure 3. A Sample of the Output

8.10. Saving the Word2Vec Tax Law Corpus Model to a File and Rotating it

We saved the trained Word2Vec model to a file using the function `model2.save('word2vec.model2')`, which stores all the model parameters, including the word vectors and training information. The saved model was later loaded using `model2 = Word2Vec.load('word2vec.model2')`. To visualise the word vectors, we created a t-SNE model with the command `tsne = TSNE(perplexity=20, random_state=0)`[4]. The perplexity parameter, a hyperparameter of the t-SNE algorithm, controls the number of neighbouring data points considered. It typically ranges between 5 and 50, with lower values focusing on smaller neighbourhoods and higher values on larger ones. In this study, the perplexity was set to 20, meaning that the algorithm considered

[4]The perplexity value in t-SNE controls the number of neighbouring data points considered, where low values focus on a smaller set of neighbours, and high values include a larger set. A value of 20 strikes a balance, providing a neighbourhood that is neither too small nor too large, effectively preserving both the local and global structure of the dataset. Typically, perplexity values between 5 and 50 yield reasonable results, with 20 being a reliable starting point for capturing the clustering structure without excessive scaling or dispersion. Practical experience suggests that a perplexity of 20 often represents the data distribution well although the optimal value can depend on factors such as dataset size and density. To ensure reproducibility, the random_state parameter fixes the randomness in t-SNE's initialisation, allowing consistent results across repeated runs with the same dataset and hyperparameters. This also aids in comparability, as using the same random_state value ensures consistent results when revisiting the same project or comparing studies in different environments. Thus, a perplexity value of 20 enables t-SNE to accurately capture the data structure and visualise cluster relationships, while a random_state value of 0 ensures that the results are repeatable and comparable.

approximately 20 neighbouring data points for each data point. The random_state parameter, acting as a stabiliser (seed), ensures that the model produces consistent results when rerun. Setting random_state=0 allowed us to achieve reproducibility, ensuring the same results in every execution. This combination of settings enabled the t-SNE algorithm to work effectively for visualising and interpreting the trained word vectors.

9. VISUALISATION

9.1. Reporting the Nearest Words as a Two-Dimensional Graph with T-SNE

Using Python functions, it is possible to create a two-dimensional graph with t-SNE (t-Distributed Stochastic Neighbour Embedding) to visualise the words closest to any word in the model.

Here, we tested our model using words as the centre of the t-SNE plot. Word vectors, initially defined as an empty NumPy array, consisted of 100-dimensional vectors per row. The word lists contained the label (name) of the centre word followed by nearby words, starting with the centre word itself. The model identified the words closest to the given word ('mükellef'), returning the most similar words along with their similarity scores. The result was a list of words and similarity scores.

The 100-dimensional vector of the centre word was extracted and added to the array. Then, the vectors of the closest words were iteratively processed, adding each word's vector to the array and its label (name) to the list. In this way, the vectors and labels of all the words were collected in a structured manner.

The perplexity value, an important t-SNE hyperparameter, was set based on the number of words. The word vectors obtained from the model were reduced to 2D space using the t-SNE algorithm, and the assigned variable Y contained the 2D coordinates of each word. The terms x_coords and y_coords represent the x and y coordinates of each word in 2D space. These coordinates were used to plot the words in 2D space on a graph.

Using the `plt.annotate()` code snippet, the label of each word was added to the graph at its respective x and y coordinates, with the labels positioned near the points. The `plt.show()` code snippet displayed the final graph on the screen.

9.2. Visualisation Command Using t-SNE (t-Distributed Stochastic Neighbour Embedding)

Finally, using the previously defined functions, a Python command was executed to visualise the closest words to the word 'mükellef' in the model via t-SNE (t-Distributed Stochastic Neighbour Embedding). This allowed the closest words to 'mükellef' to be graphically displayed, as shown in Figure 3.

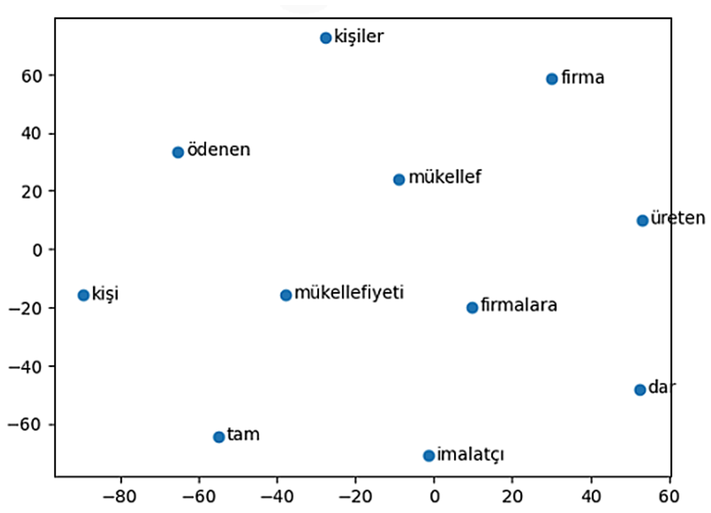

Figure 3. Words Most Similar to the Word 'mükellef'

According to Article 8/1 of the Tax Procedure Law (Vergi Usul Kanunu), "a taxpayer is a natural or legal person to whom a tax obligation is imposed according to tax law" (mükellef, vergi kanunlarına göre kendisine vergi borcu terettübeden gerçek veya tüzel kişidir). To "hesitate" (tereddüb etmek) means to have a duty or to be required (üzerine görev düşmek, gerekme anlamına gelir) [24]. Taxpayers are referred to as 'kişi' (real or legal persons) or 'şirket' (companies) in various parts of the law.

Mükellefler (taxpayers) are defined in Articles 3–6 of the Income Tax Law and Articles 3–6 of the Corporate Tax Law. In these articles, taxpayers are divided into two different classes: 'tam' and 'dar'. These persons often assume the identity of the producers and manufacturers. From this perspective, it is evident that the words associated with 'mükellef' in the table possess characteristics that align with the concept of 'mükellef', demonstrating that the similarity model is functioning effectively.

10. CONCLUSION AND EVALUATION

In this paper, the Word2Vec model was used to train a model tailored to the legal context by working on a corpus specific to Turkish tax laws. Word2Vec represents words as vectors and identifies semantic relationships between them based on the proximity of these vectors in vector space. The analysis focused on the connections between words found in tax laws and other related terms. Using Python libraries such as NumPy, Gensim, Scikit-learn, and Matplotlib, high-dimensional data was visualised on a two-dimensional plane through the t-SNE algorithm. This method made the relationships between similar terms used in tax legislation observable and provided a deeper understanding of their meanings in the context of tax laws, leveraging machine learning techniques.

With over 25 years of professional experience in tax law and a doctorate in the field, I can confidently state that, in the context of the word "mükellef" (taxpayer), the relationships between words in the laws are meaningful and consistent. Similarly, the approach yielded successful results for other tax-related terms not included in this paper.

The word vectors generated by the Word2Vec model offer a powerful foundation for understanding the semantic affinities of terms in tax laws. For instance, the semantic proximity of the word "mükellef" to terms such as "tam" (full), "dar" (narrow), "şirket" (company), and "üretici" (manufacturer) demonstrates how key concepts in tax law are interrelated. Such analyses facilitate the grouping of terms with similar meanings and provide a clearer understanding of the conceptual basis of legal regulations. The t-SNE visualisation simplifies the relationships between words in tax laws by reducing them to a two-dimensional plane, making it easier to observe how words cluster in a legal context. These visualisations can serve as a foundation for legal decision support systems, enabling automated legal advice based on visualised word relationships.

Vector relationships offer an excellent primer for AI-assisted tax law advice, enabling AI systems to become more effective and context-sensitive. The dense and technical nature of legal language makes comprehension difficult, but AI models can play a significant role in simplifying legal texts, enhancing transparency and accessibility in legal processes. Similarity analysis and the use of word vectors can expedite relationships between legal documents and streamline processes, particularly in litigation or legal review scenarios. A correct understanding of tax law terminology allows AI consultancy systems to respond to user requests with greater accuracy.

Natural language processing models like Word2Vec interpret the words in tax laws, learn the relationships between them, and make complex regulatory information analysable through machine learning. This accelerates the delivery of information about tax legislation while improving accuracy. Future developments of this study, such as the use of larger datasets and different NLP models, could enable the creation of user-friendly consultancy platforms that provide in-depth, reliable information on tax law and adapt quickly to legal changes.

The methodology in this paper needs further refinement to model more complex legal relationships and to highlight rare but critical words. However, the potential of the model presented here as a foundation for AI assistants in tax law consultancy is noteworthy. Developing a tax law-specific Word2Vec model enables the creation of AI solutions that are more customised and context-sensitiver than general-purpose language models. This enhances the ability of AI-based applications to draw meaningful conclusions from legal documents, improves the accuracy of information provided to users, and minimizes errors in legal processes. Such systems can elevate legal advice services by delivering more accurate and consistent information tailored to users' needs.

This infrastructure paves the way for innovation in the sector, enabling tax law AI assistants to offer reliable, detailed recommendations tailored to taxpayers. Future research could explore comparisons with other word representation methods, integration with context-aware models like BERT or GPT, summarisation of legal texts, application to real-world tax disputes, and the development of automated response systems (ChatBots).

Peer Review Externally peer-reviewed.
Conflict of Interest The author has no conflict of interest to declare.
Grant Support The author declared that this study has received no financial support.

Author Details
Ali İhsan Özgür Çilingir
[1] Non-affiliated, İstanbul, Türkiye
0000-0002-0490-4192

References

[1] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient Estimation of Word Representations on Vector Space. arXiv preprint arXiv:1301.3781 (2013).

[2] Pervan, Nergis. DERİN ÖĞRENME YAKLAŞIMLARI KULLANARAK TÜRKÇE METİNLERDEN ANLAMSAL ÇIKARIM YAPMA. Ankara, 2019.

[3] Onan, Aytuğ. Evrişimli Sinir Ağı Mimarilerine Dayalı Türkçe Duygu Analizi. Avrupa Bilim ve Teknoloji Dergisi (Aug. 31, 2020), 374-380.

[4] Tezgider, Murat, Yıldız, Beytullah, and Aydın, Galip. Improving Word Representation by Tuning Word2Vec Parameters with Deep Learning Model. In International Artificial Intelligence and Data Processing Symposium (IDAP) (Malatya 2018), IEEE, 1-7.

[5] Arabacı, Mehmet Ali, Esen, Ersin, Atar, Muhammed Selim, Yılmaz, Eyüp, and Kaltalıoğlu, Batuhan. Kelime Gömevi Yöntemi Kullanarak Benzer Cümle Tespiti. In 2018 26th Signal Processing and Communications Applications Conference (2018).

[6] Aydoğan, Murat and Karcı, Ali. Kelime Temsil Yöntemleri ile Kelime Benzerliklerinin İncelenmesi. Çukurova Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 34, 2 (June 2019), 181-195.

[7] Acı, Çiğdem İnan and Çırak, Adem. Türkçe Haber Metinlerinin Konvolüsyonel Sinir Ağları ve Word2Vec Kullanılarak Sınıflandırılması. BİLİŞİM TEKNOLOJİLERİ DERGİSİ, 12, 13 (July 31, 2019), 219-228.

[8] Xia, Chunyu, He, Tieke, Li, Wenlong, Qin, Zemin, and Zou, Zhipeng. Similarity Analysis of Law Documents Based on Word2vec. In 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (Sofia 2019), IEEE, 354-357.

[9] Tatchum, Ghislain Wabo, Makembe, Fritz Sosso, Nzeko, Armel Jacques Nzekon, and Djam, Xaviera Youh. Class-Oriented Text Vectorization for Text Classification: Case Study of Job Offer Classification. Journal of Computer Science an Engineering (JCSE), 5, 2 (Aug. 01, 2024), 116-136.

[10] Wei, Wei, Liu, Wei, Zhang, Beibei, Scherer, Rafal, and Damasevicius, Robertas. Discovery of New Words in Tax-related Fields Based on Word Vector Representation. Journal of Internet Technology, 24, 4 (July 2023), 923-930.

[11] Chalkidis, Ilias and Kampas, Dimitrios. Deep learning in law: early adaptation and legal word embeddings trained on large corpora. Artificial Intelligence and Law ((Dec. 2019), 171-198.

[12] Mandal, Arpan, Ghosh, Kripabandhu, Ghosh, Saptarshi, and Mandal, Sekhar. Unsupervised approaches for measuring textual similarity between legal court case reports. Artificial Intelligence and Law, 29 (2021), 417-451.

[13] Saha, Rohan. Influence of various text embeddings on clustering performance in NLP. arXiv, 44 (May 04, 2023), 1-22.

[14] Zhong, Ziyuan, Verma, Nakul, and Lia, Vincent. Lecture 8 – t-Distributed Stochastic Neighbor Embedding. New York, 2018.

[15] Linderman, George C. and Steinerberger, Stefan. CLUSTERING WITH T-SNE, PROVABLY. arXiv (June 08, 2017), 1-15.

[16] Arora, Sanjeev and Hu, Wei. An Analysis of the t-SNE Algorithm for Data Visualization. In Conference on Learning Theory (COLT) 2018 (Stockholm 2018), arXiv, 1-32.

[17] Maaten, Laurens van der and Hinton, Geoffrey. Visualizing Data using t-SNE. Journal of Machine Learning Research, 9, 86 (Sep. 2008), 2579–2605.

[18] Mueller, John Paul and Massaron, Luca. Deep Learning for Dummies. John Wiley & Sons, Inc., New Jersey, 2019.

[19] Nelson, Hala. Essential Math for AI - Next Level Mathematics for Efficient and Succesful AI Systems. O'Reilly Media, Sebastopol, 2023.

[20] Kelleher, John D. Deep Learning. The MIT Press, London, 2019.

[21] Anonymous. NumPy documentation. 2024.

[22] https://scikit-learn.org/stable/. https://scikit-learn.org/stable/. 2024.

[23] https://matplotlib.org/. https://matplotlib.org/. 2024.

[24] Anonim. Osmanlı Türkçesi Sözlüğü.

[25] Haider, Mofiz Mojib, Hossin, Arman, Mahi, Hasibur Rashid, and Arif, Hossain. Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm. In 2020 IEEE Region 10 Symposium (TENSYMP) (Dhaka 2020), 283-286.

[26] Li, Zhie and Rao, Zhuyi. Text classification model based on Word2vec and SF-HAN. In 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC 2020) (Shenzhen 2020), 978-1-7281-4323-1/20/$31.00 ©2020 IEEE, 1385-1390.

[27] Mao, Yushang, Zhang, Guixuan, and Zhang, Shuwu. Word Semantic Similarity Based on CiLin and Word2vec. In 2020 International Conference on Culture-oriented Science & Technology (ICCST) (Beijing), 978-1-7281-8138-7/20/$31.00 ©2020 IEEE, 304 - 307.

[28] Bissiri, Pier Giovanni and Walker, Stephen G. Converting information into probability measures with the Kullback–Leibler divergence. Ann Inst Stat Math (2012), 1139-1160.

[29] Jaya, Putra Syopiansyah, Nur, Gunawan Muhamad, and Akbar, Hidayat Arief. Feature Engineering with Word2vec on Text Classification Using The K-Nearest Neighbor Algorithm. In The 10th International Conference on Cyber and IT Service Management (CITSM 2022) (Yogyakarta 2022), ©2022 IEEE.

[30] Kurian, Jeomoan Francis and Allali, Mohamed. Detecting drifts in data streams using Kullback-Leibler (KL) divergence measure for data engineering applications. Journal of Data, Information and Management (2024), 207-2016.

[31] Polat, Buğra. TÜRKÇE ÜRÜN YORUMLARI VERİSİ İLE DUYGU ANALİZİ. Ankara, 2021.

[32] Çalışkan, Sedrettin, Yazıcıoğlu, Selahattin A., Demirci, Ulaş, and Kuş, Zeki. YAPAY SİNİR AĞLARI, KELİME VEKTÖRLERİ VE DERİN ÖĞRENME UYGULAMALARI. İstanbul, 2018.

[33] Pirana, Gurur, Sertbaş, Ahmet, and Ensari, Tolga. Sanal Asistan Uygulamaları İçin Derin Öğrenme Yöntemiyle Cümle Sınıflandırma. In 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT 2019 (Ankara 2019), Institute of Electrical and Electronics Engineers Inc.

[34] Kılıç, Berker and Öner, Yüksel. Yargıtay Kararlarının Suç Türlerine Göre Makine Öğrenmesi Yöntemleri İle Sınıflandırılması. VERİ BİLİMİ DERGİSİ (2021), 61-71.

[35] Law, Jarvan, Zhuo, Hankui Hankz, He, Junhua, and Rong, Erhu. LTSG: Latent Topical Skip-Gram for Mutually Learning Topic Model and Vector Representations. arXiv preprint arXiv (Feb. 23, 2017).

[36] Önal, Zeynep. Derin Öğrenme. Nobel Akademik Yayıncılık, Ankara, 2022.

[37] Guthrie, David, Allison, Ben, Liu, Wei, Guthiere, Louise, and Wilks, Yorick. A Closer Look at Skip-gram Modelling. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06) (Genoa 2006), ACL Anthology, 1222-1225.

[38] Srivastava, Rajendra P. New Measure of Similarity in Textual Analysis: Vector Similarity Metric versus Cosine Similarity Metric. JOURNAL OF EMERGING TECHNOLOGIES IN ACCOUNTING, 20, 1 (2023), 77-90.

[39] Pudaruth, Sameerchand, Soyjaudah, Sunjiv, and Gunputh, Rajendra. Classification of Legislations using Deep Learning. The International Arab Journal of Information Technology, 18, 5 (Sep. 2021), 651-663.

[40] Robaldo, Livio, Villiata, Serena, Wyner, Adam, and Grabmair, Matthias. Introduction for artificial intelligence and law: special issue "natural language processing for legal texts". Artificial Intelligence and Law (Apr. 2019), 113-115.

[41] Tagarelli, Andrea and Simeri, Andrea. Unsupervised law article mining based on deep pre-trained language representation models with application to the Italian civil code. Artificial Intelligence and Law, 30 (Sep. 2022), 417-473.

[42] Makawana, Mayur and Mehta, Rupa G. A novel network-based paragraph filtering technique for legal document similarity analysis. Artificial Intelligence and Law (Oct. 2023).

[43] Bilgin, Metin. Kelime Vektörü Yöntemlerinin Model Oluşturma Sürelerinin Karşılaştırılması. BİLİŞİM TEKNOLOJİLERİ DERGİSİ, 12, 2 (Apr. 2019), 141-146.

[44] Ahmetoğlu, Hüseyin and Daş, Resul. Türkçe Otel Yorumlarıyla E˘gitilen Kelime Vektörü Modellerinin Duygu Analizi ile ˙Incelenmesi. Fen Bilimleri Enstitüsü Dergisi, 24, 2 (2020), 455-463.

[45] Çelik, Özer and Koç, Burak Can. TF-IDF, Word2vec ve Fasttext Vektör Model Yöntemleri ile Türkçe Haber Metinlerinin Sınıflandırılması. Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi, 23 (2021), 121-127.

[46] Kınık, Doğancan and Güran, Aysun. TF-IDF ve Doc2Vec Tabanlı Türkçe Metin Sınıflandırma Sisteminin Başarım Değerinin Ardışık Kelime Grubu Tespiti ile Arttırılması. Avrupa Bilim ve Teknoloji Dergisi (Jan. 2021), 323-332.

[47] Hongnan, Tian and Xin, Guo. Research on Improved Sentence Similarity Calculation Method Based on Word2Vec and Synonym Table in Interactive Machine Translation. In 2021 5th International Conference on Robotics and Automation Sciences (Wuhan 2021), IEEE , 255-261.

[48] Xiao, Lu, Li, Qiaoxing, Ma, Qian, Shen, Jiasheng, Yang, Yong, and Li, Danyang. Text classification algorithm of tourist attractions subcategories with modified TF-IDF. PLOS ONE (Oct. 2024), 1-34.

[49] Gupta, Megha, Dheekonda, Venkatasai, and Masum, Mohammad. Genie: Enhancing information management in the restaurant industry through AI-powered chatbot. International Journal of Information Management Data Insights (May 25, 2024), 1-9.

[50] G, Dhamodharan and A, Kaleemullah. An Innovative Algorithm for Enhanced PDF-Based Chatbot in Domain-Specific Question Answering. Library Progress International, 44, 3 (Sep. 01, 2024), 27648-27653.

[51] Godghase, Gauri Anil, Agrawal, Rishit, Obili, Tanush, and Stamp, Mark. Distinguishing Chatbot from Human. arXiv:2408.04647v1 [cs.CL] (Aug. 12, 2024), 1-47.

[52] Becha, Rahma, Sellami, Asma, Bouassida, Nadia, Idri, Ali, and Abran, Alain. BotCFP: A Machine Learning based Tool for COSMIC Chatbots Sizing. CEUR, 3852 (Apr. 30, 2024), 1-16.

[53] https://pypi.org/project/gensim/. https://pypi.org/project/gensim/. 2024.

[54] Leshem, Ido. Skip-Gram Word2Vec Algorithm Explained. 2023.