2024 Innovations in Intelligent Systems and Applications Conference (ASYU) | 979-8-3503-7943-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASYU62119.2024.10757157

Applying Weighted Graph Embeddings to Turkish Metaphor Detection

Emrah Inan
Computer Engineering
Izmir Institute of Technology
35430 Urla, Izmir, Turkiye
emrahinan@iyte.edu.tr

***Abstract*—Metaphor is a common literary mechanism that allows abstract concepts to be conceptualised using more concrete terminology. Existing methods rely on either end-to-end models or hand-crafted pre-processing steps. Generating well-defined training datasets for supervised models is a time-consuming operation for this type of problem. There is also a lack of pre-processing steps for resource-poor natural languages. In this study, we propose an approach for detecting Turkish metaphorical concepts. Initially, we collect non-literal concepts including their meaning and reference sentences by employing a Turkish dictionary. Secondly, we generate a graph by discovering super-sense relations between sample texts including target metaphorical expressions in Turkish WordNet. We also compute weights for relations based on the path closeness and word occurrences. Finally, we classify the texts by leveraging a weighted graph embedding model. The evaluation setup indicates that the proposed approach reaches the best F1 and Gmean scores of 0.83 and 0.68 for the generated test sets when we use feature vector representations of the Node2Vec model as the input of the logistic regression for detecting metaphors in Turkish texts.**

***Index Terms*—metaphor detection, node2vec model, Turkish, metaphor dataset**

I. Introduction

Metaphor is a common property of human language that allows abstract concepts to be conceptualised using more concrete terms. In this case, a source expression can be used to represent a view of a target expression in a conceptual metaphor mapping [1]. As denoted in a recent study [2], linguistic rules facilitate the identification of metaphors. A metaphor can be identified when there is a more concrete or historically older contrastive meaning for the target word, which tends to be identified by the Metaphor Identification Procedure (MIP). On the other hand, Selectional Preference Violation (SPV) is another linguistic rule which states that a metaphor is identified by noting the semantic mismatch between a target word and its context.

Language models improve the performance of the metaphor identification and the detection tasks. MelBERT [3] detects metaphorical usage by a simple chunking method using comma-separated sub-sentences consisting of a target word represented by a special symbol. RoPPT [4] detects metaphors by using RoBERTa based on a goal-oriented parse tree structure.

To reduce the expense of the fine-tuning process, a recent graph-based approach [5] covers the dependency parse tree of texts and contextual clues gathered from neighbouring nodes. It detects a context-inclusive metaphor using a linear SVM model. However, there is a lack of high performance open source dependency parser libraries for low resource languages. This work proposes a Node2Vec [6]-based approach that does not require a dependency parser library to detect metaphors.

The main contributions of our work can be summarised as follows: First, we generate a Turkish metaphor detection dataset from the Turkish TDK dictionary for training and test cases. We map the Turkish WordNet [7] to the generated metaphor dataset by using the exact string matching algorithm and Jaccard similarity for the definitions of concepts in WordNet and target word meanings from the dictionary to build the metaphor graph. Further, we also employ Node2Vec [6] on the generated graph. We perform the Logistic Regression method to classify the given text as literal or metaphor. The organization of the paper is structured as follows. Section 2 introduces related work. Section 3 presents the proposed graph-based metaphor detection method. Section 4 proposes the evaluation of the dataset and the experiments. Finally, Section 5 concludes the study and states the future directions.

II. Related Work

The detection of metaphors is usually conceived as a task of either metaphorical or literal use in binary classifiers based on triples of subjects, verbs and objects or on entire sample texts. Neuman et al. [8] identify a Concrete Category Overlap algorithm that uses the co-occurrence statistics and abstractness scores to state WordNet [9] super-senses as an indicator of literal usage. Heintz et al. [10] propose a system for English and Spanish metaphor detection based on LDA topic modelling that enables it applicable to low-resource languages without requiring labelled data. Wilks et al. [11] describe a method using WordNet where metaphors are coded into the senses.

Gandy et al. [12] propose a method that detects linguistic metaphors and clusters them to find conceptual metaphors. Their method leverages lexical semantics in the form of Wordnet and Wiktionary, and generic syntactic parsing and part-of-speech tagging as minimal background knowledge. Mohler et al. [13] present an example-based approach to detect metaphor by comparing the domain-aware semantic signature of a text with a large index of known metaphors. Tsvetkov et al. [14] propose a language-independent method in which they employ affective ratings, WordNet categories and vector-space word representations to detect multilingual target terms in English, Russian, Spanish and Farsi. Köper and im Walde [15] present a binary and token-based metaphor detection method for German particle verbs by using a random forest classifier that incorporates particle-verb specific features and noun clusters.

Lin et al. [16] introduce a mixed method of semi-supervised learning with self-training to augment an unlabelled metaphor dataset, where they incrementally add unlabelled data during the training phase and generate pseudo-labels based on a contrastive learning objective to measure the distance between literal and metaphorical meanings. Qin and Zhao [17] propose a Transformer-based neural network model by extracting frequently associated subjects and objects of the target words from Wikipedia. The vector representations of the original and extracted subjects and objects were considered as the contrast between metaphorical and literal meanings.

III. Method

Our method consists of three main tasks, including metaphor graph generation, Node2Vec representation from the generated graph, and metaphor detection using the logistic regression method. Figure 1 shows the general structure of the proposed method. To generate the metaphor graph, we employ both the Turkish WordNet [7] and the generated metaphor dataset from the Turkish dictionary. After that, we represent each node of the generated graph as a vector representation with the help of Node2Vec. Finally, we perform the logistic regression method to classify the given text as literal or metaphor.

A. Metaphor graph

We generate the metaphor graph from sample sentences for each target word extracted from the Turkish dictionary and we map them the corresponding concepts to the Turkish WordNet. Figure 2 illustrates a sample visualisation of the generated graph.

Initially, we gather sample texts and meaning of each target word from the online Turkish dictionary. To guarantee the mapping between the target words and the concepts of the Turkish WordNet, we extract only dictionary items for each concept in the WordNet. The dashed connection between the concept c1 and the target word w1 indicates that these terms are the same term. We use the exact string matching method and compute the Jaccard similarity score of the meaning of the target term and the definition of the concept in the WordNet.

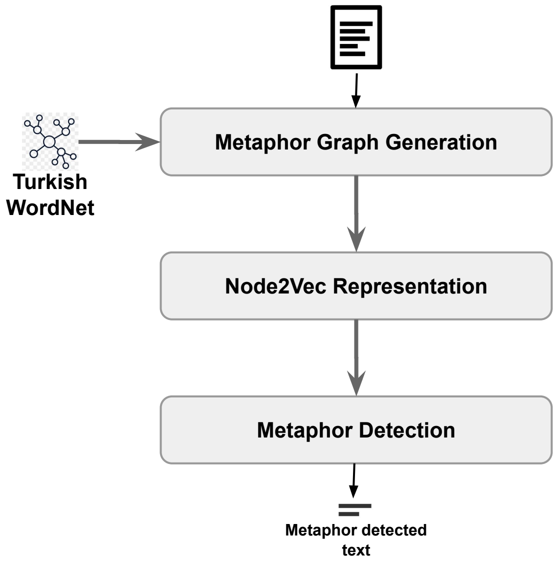

Fig. 1. General structure of the proposed method.

To generate subgraphs from sample sentences, we first remove punctuation and stop words. Then we lemmatise the target and context words cw_i by using the NLTK toolkit. If target and context words are taken place in the same sample text, we link them together. We also check whether there exists a word co-occurrence between sample texts, then we connect these nodes with dotted edges. For instance, the context words cw_3 and cw_4 are the same words and they also exist in the second sample text.

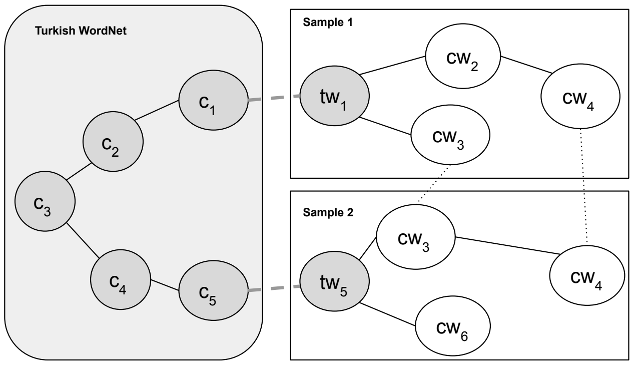

Fig. 2. An example for the generated metaphor graph.

B. Node2Vec

The Node2Vec [6] method is employed for learning continuous feature representations for nodes in general networks. The method is based on the node mappings to a low-dimensional space of features that maximises the likelihood by preserving the neighbourhoods of the node structure. The method explores diverse neighbourhoods of nodes by using a biased random walk which is parameterised to follow a specific concept of the node neighbourhood.

The Node2Vec method is a 2-step representation learning algorithm. These steps are (1) It employs a second order random walk to obtain sentence representations as node ids from the given graph. (2) It then uses these representations to train an embedding layer for each vector for each node in the graph. After that, it leverages Word2Vec [18] for computing the embedding vectors. Given β as the source node and n_i as the ith node in the random walk, beginning with $n_0 = \beta$, nodes n_i are generated as follows:

$$P(n_i = x|n_i - 1 = v) = \begin{cases} \frac{\pi_{vx}}{K}, & \text{if } (v, x) \in E \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where π_{vx} is the unnormalised transition probability between nodes x and v. E is the edge list and K is the normalising constant. It uses the transition probabilities π_{vx} on edges (x, v). Then, we set the transition probability as

$$\pi_{vx} = \alpha_{pq}(t, x).w_{vx} \tag{2}$$

where,

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p}, & \text{if } d_{tx} = 0 \\ 1, & \text{if } d_{tx} = 1 \\ \frac{1}{q}, & \text{if } d_{tx} = 2 \end{cases} \tag{3}$$

where d_{tx} indicates the shortest path distance between nodes t and x. In the case of unweighted graphs, the static edge weights w_{vx} are set to 1. We set the parameters p and q to 0.5 and 2.0 respectively for fixed maximum length of random walks. For each node, we generate 10 random walks in the graph with a length of up to 100. To define edge weights for weighted random walks of the underlying graph, we compute weights for each edge by the similarity of its end nodes. We assign edge weights by using the Jaccard similarity of the features for node pairs as one-hot coded features.

C. Metaphor detection

After obtaining the sample set of random walks and their weights, we employ the Word2Vec model in Gensim to learn the low-dimensional embedding of the nodes. We set the dimensionality of the learned embedding vectors to 128. Since we compute the node embeddings using Word2Vec, we use these embeddings as feature vectors in the metaphor detection task. Finally, we use the logistic regression method from the Scikit-learn library as a classifier of the given text.

$$l = \log \frac{p = P(Y = 1)}{1 - (p = 1)} = \beta_0 + \beta_1.x_1 + +\beta_2.x_2 + ... + +\beta_N.x_N \tag{4}$$

where β is the parameter of the model, x is the predictor, N is the size of the predictors and Y is the binary response variable denoted as $p = P(Y = 1)$.

IV. Evaluation

To evaluate our method, we randomly extract a test dataset from each generated dataset into 500 metaphorical and 500 literal tuples consisting of term, meaning type and sample texts.

A. Dataset generation

To generate an evaluation dataset, we employ the Jsoup library to extract all the terms from the Turkish Wiktionary, starting from A to Z characters. We get the word by querying the Turkish dictionary for each concept and get the meaning, the type as a literal or metaphorical label and the sentence containing the given concept. Table 1 shows the details of the generated dataset.

TABLE I
Statistics of the generated dataset.

Dataset	Metaphorical	Literal
UnSaCle	2337	18905
UnSaMeCle	4674	37810
BaSaCle	2337	2337
BaSaMeCle	4674	4674

The first dataset contains 2337 metaphorical and 19200 literal sample texts with their features for Unbalanced Sample Clean (UnSaCle), where the feature representation includes only sample texts of terms. Unbalanced Sample Meaning Clean (UnSaMeCle) contains both sample and meaning of terms and the number of metaphorical and literal texts is 2 times larger than the UnSaCle dataset. To transform these datasets, we extract the same number of texts for both types of terms and then we obtain the Balanced SampleClean (BaSaCle) dataset which contains 2337 texts for both terms. Similarly, we also contain Balanced Sample Meaning Clean (BaSaMeCle) dataset which also includes sample and meaning of these terms.

TABLE II
Generated examples for the concept "koku (smell)".

Term	Lang.	Meaning	Type	Sample text
Koku	Tr	Nesnelerden yayılan küçücük zerrelerin burun zarı üzerindeki özel sinirlerde uyandığı duygu	Gerçek	Çöp kokusuyla beraber mutsuzluğu da artıyordu günbegün.
Smell	En	Feelings of tiny particles emanating from objects evoke special nerves on the nasal membrane	Literal	With the smell of garbage, his unhappiness was increasing day by day.
Koku	Tr	Belirti, işaret	Mecaz	Ortalıkta bir savaş kokusu var.
Smell	En	Glimpse, sign	Metaphor	There is a smell of war around.

For instance, we extract the information about the Turkish term "koku (smell)" as illustrated in Table 2. It shows the sample concept with the collected knowledge from the Turkish dictionary (experts have already assigned labels, either literally or metaphorically) with its English translation. The actual reference of the term "smell" means "smell of a garbage" in Turkish. On the other side, it can also be a glimpse or a sign of an event as a metaphorical information. By extracting these terms we obtain unbalanced and balanced datasets. For each dataset, we also generate feature representations by using sample texts and the meaning of terms.

B. Results

We perform experiments using the implementations of Logistic Regression (LoRe), SVM and Random Forest (RaFo) classifiers in the Scikit-learn library. For all supervised learning methods, we utilise the default settings of Scikit-learn and we apply these methods to the pipeline for the transformation with a final estimator. A pipeline from Scikit-learn requires steps including a fit/transform chain with the last object being an estimator. As transformers, we perform Doc2Vec, TF-IDF and Unigram feature representations and we examine LoRe, SVM and RaFo methods as final estimators.

TABLE III
COMPARISON OF METHODS ON UNSAMECLE DATASET.

Method	P	R	F1	GMean
LoRe$_{\textbf{Node2Vec}}$	**0.95**	**0.74**	**0.83**	**0.68**
$LoRe_{Node2VecWeighted}$	0.93	0.73	0.82	0.64
$LoRe_{Doc2Vec}$	0.93	0.70	0.80	0.61
$SVM_{Doc2Vec}$	0.91	0.55	0.68	0.58
$Rafo_{Doc2Vec}$	0.90	0.61	0.73	0.57
$LoRe_{TF-IDF}$	0.93	0.72	0.81	0.62
SVM_{TF-IDF}	0.89	0.48	0.63	0.51
$Rafo_{TF-IDF}$	0.90	0.53	0.67	0.53
$LoRe_{Unigram}$	0.90	0.46	0.54	0.47
$SVM_{Unigram}$	0.91	0.33	0.48	0.49
$Rafo_{Unigram}$	0.92	0.32	0.47	0.49

For the experimental setup, we use 10-fold cross-validation as a common strategy for classifier performance estimation. Hence, we again employ Scikit-learn to split each dataset into 10 blocks. We randomly retain one single block as the validation data for testing the given method and we train this method on the remaining 9 blocks. We repeat the cross-validation process 10 times and reach the accuracy results for the methods.

TABLE IV
COMPARISON OF METHODS ON UNSACLE DATASET.

Method	P	R	F1	GMean
LoRe$_{\textbf{Node2Vec}}$	**0.91**	**0.62**	**0.74**	**0.57**
$LoRe_{Node2VecWeighted}$	0.90	0.61	0.73	0.55
$LoRe_{Doc2Vec}$	0.89	0.63	0.74	0.56
$SVM_{Doc2Vec}$	0.88	0.62	0.73	0.54
$Rafo_{Doc2Vec}$	0.88	0.61	0.72	0.54
$LoRe_{TF-IDF}$	0.89	0.58	0.70	0.52
SVM_{TF-IDF}	0.87	0.56	0.68	0.51
$Rafo_{TF-IDF}$	0.88	0.58	0.70	0.51
$LoRe_{Unigram}$	0.86	0.57	0.68	0.51
$SVM_{Unigram}$	0.87	0.56	0.68	0.49
$Rafo_{Unigram}$	0.85	0.56	0.67	0.50

TABLE V
COMPARISON OF METHODS ON BASAMECLE DATASET.

Method	P	R	F1	GMean
$LoRe_{Node2Vec}$	0.64	0.53	0.58	0.59
$LoRe_{Node2VecWeighted}$	0.66	0.59	0.62	0.64
$LoRe_{Doc2Vec}$	0.67	0.60	0.63	0.64
$SVM_{Doc2Vec}$	0.66	0.43	0.52	0.57
$Rafo_{Doc2Vec}$	0.67	0.61	0.64	0.65
LoRe$_{\textbf{TF-IDF}}$	**0.70**	**0.62**	**0.66**	**0.67**
SVM_{TF-IDF}	0.69	0.61	0.65	0.66
$Rafo_{TF-IDF}$	0.68	0.60	0.64	0.65
$LoRe_{Unigram}$	0.65	0.59	0.62	0.64
$SVM_{Unigram}$	0.63	0.57	0.60	0.62
$Rafo_{Unigram}$	0.61	0.62	0.61	0.60

Table 3 shows the Precision (P), Recall (R), F1 and GMean (suitable for unbalanced datasets) results for the supervised learning methods depending on different feature vector representations such as Doc2Vec, TF-IDF, Unigram and Node2Vec models. When the Doc2Vec model is selected as the feature representation, Logistic Regression (LoRe) gives better results than other methods for TF-IDF and Unigram feature vectors. Our first method, based on the unweighted Node2Vec method, performs better than all other methods. In this unweighted method, every edge has equal importance and set to 1. Table 4 denotes the comparison results considering only sample texts of the anchor terms. Table 5 illustrates the comparison results on the balanced dataset for both the sample and the meaning of the anchor terms.

Table 5 shows the results of the balanced sample size and definition of the given words. For smaller datasets, the Unigram and TF-IDF based methods perform better, as the Node2Vec and Doc2Vec models tend to require a large number of records. Table 6 shows the comparison results on the balanced dataset for the example texts containing only the anchor terms.

TABLE VI
COMPARISON OF METHODS ON BASACLE DATASET.

Method	P	R	F1	GMean
$LoRe_{Node2Vec}$	0.57	0.55	0.56	0.57
$LoRe_{Node2VecWeighted}$	0.59	0.52	0.55	0.57
$LoRe_{Doc2Vec}$	0.57	0.54	0.55	0.58
$SVM_{Doc2Vec}$	0.59	0.54	0.56	0.58
$Rafo_{Doc2Vec}$	0.59	0.57	0.58	0.59
$LoRe_{TF-IDF}$	0.58	0.40	0.47	0.52
SVM_{TF-IDF}	0.57	0.40	0.47	0.52
$Rafo_{TF-IDF}$	0.58	0.43	0.49	0.53
$LoRe_{Unigram}$	0.56	0.56	0.56	0.55
$SVM_{Unigram}$	0.55	0.54	0.54	0.55
Rafo$_{\textbf{Unigram}}$	**0.54**	**0.70**	**0.61**	**0.54**

Precision results are generally higher in these tables, since literal representations are common in the literature, and hence these methods tend to predict literal target words that are actually metaphorical entries. Lastly, in Table 7, we compare path-based similarity methods, including Resnik [19], JCN [20], LCH [21], WuPalmer [22] and Lin [23] and We compute edge weights with these methods for weighted Node2Vec representations. Resnik similarity [19] concentrates on the sim-

TABLE VII
COMPARISON OF SIMILARITY METHODS FOR WEIGHTED NODE2VEC REPRESETATIONS.

Similarity	**Acc$_{\text{UnSaCle}}$**	**Acc$_{\text{UnSaMeCle}}$**
Resnik [19]	0.909	0.918
JCN [20]	0.912	0.925
LCH [21]	0.91	0.924
WuPalmer [22]	0.907	0.916
Lin [23]	**0.916**	**0.928**

ilarity between two synsets based on the Information Content (IC) of the most specific ancestral node. JCN similarity [20] leverages the lowest common parent of two synsets in the same taxonomy. LCH similarity [21] relies on the shortest path between the hypernym/hyponym taxonomy of two synsets and its maximum depth. WuPalmer [22] method depends on the depth of the two nodes in the WordNet and the depth of their most specific ancestors. Lin [23] method relies on both the information needed to state the commonality between two synsets and the descriptions.

V. Conclusion

In this paper we present a method to distinguish metaphorical from non-figurative representations of words using a Node2Vec model. Experiments indicate that our method reaches remarkable results when we use feature vector representations of the Node2Vec model as input to logistic regression to detect metaphors in Turkish texts.

Future work will include the adaptation of the metaphor detection task to the psychological experiments in the analysis of individual differences [24]. As another future direction, we will investigate multimodal metaphor detection [25] by capturing the bidirectional interactions between textual and visual metaphorical features and a multi-interaction cross-modal feature fusion mechanism.

References

[1] G. Lakoff and M. Johnson, *Metaphors we live by*. University of Chicago press, 2008.

[2] S. Zhang and Y. Liu, “Metaphor detection via linguistics enhanced siamese network,” in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 4149–4159.

[3] M. Choi, S. Lee, E. Choi, H. Park, J. Lee, D. Lee, and J. Lee, “Melbert: Metaphor detection via contextualized late interaction using metaphorical identification theories,” in *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[4] S. Wang, Y. Li, C. Lin, L. Barrault, and F. Guerin, “Metaphor detection with effective context denoising,” in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 1404–1409.

[5] S. Rai, S. Chakraverty, D. K. Tayal, and Y. Kukreti, “A study on impact of context on metaphor detection,” *The Computer Journal*, vol. 61, no. 11, pp. 1667–1682, 2018.

[6] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[7] R. Ehsani, E. Solak, and O. T. Yildiz, “Constructing a wordnet for turkish using manual and automatic annotation,” *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 17, no. 3, pp. 1–15, 2018.

[8] Y. Neuman, D. Assaf, Y. Cohen, M. Last, S. Argamon, N. Howard, and O. Frieder, “Metaphor identification in large texts corpora,” *PloS one*, vol. 8, no. 4, p. e62343, 2013.

[9] C. Fellbaum, “Wordnet,” *The encyclopedia of applied linguistics*, 2012.

[10] I. Heintz, R. Gabbard, M. Srivastava, D. Barner, D. Black, M. Friedman, and R. Weischedel, “Automatic extraction of linguistic metaphors with lda topic modeling,” in *Proceedings of the First Workshop on Metaphor in NLP*, 2013, pp. 58–66.

[11] Y. Wilks, A. Dalton, J. Allen, and L. Galescu, “Automatic metaphor detection using large-scale lexical resources and conventional metaphor extraction,” in *Proceedings of the First Workshop on Metaphor in NLP*, 2013, pp. 36–44.

[12] L. Gandy, N. Allan, M. Atallah, O. Frieder, N. Howard, S. Kanareykin, M. Koppel, M. Last, Y. Neuman, and S. Argamon, “Automatic identification of conceptual metaphors with limited knowledge,” in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[13] M. Mohler, D. Bracewell, M. Tomlinson, and D. Hinote, “Semantic signatures for example-based linguistic metaphor detection,” in *Proceedings of the First Workshop on Metaphor in NLP*, 2013, pp. 27–35.

[14] Y. Tsvetkov, L. Boytsov, A. Gershman, E. Nyberg, and C. Dyer, “Metaphor detection with cross-lingual model transfer,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 248–258.

[15] M. Köper and S. S. im Walde, “Distinguishing literal and non-literal usage of german particle verbs,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 353–362.

[16] Z. Lin, Q. Ma, J. Yan, and J. Chen, “Cate: A contrastive pre-trained model for metaphor detection with semi-supervised learning,” in *Proceedings of the 2021 conference on empirical methods in natural language processing*, 2021, pp. 3888–3898.

[17] W. Qin and D. Zhao, “Background semantic information improves verbal metaphor identification,” in *Natural Language Processing and Chinese Computing: 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13–17, 2021, Proceedings, Part II 10*. Springer, 2021, pp. 288–300.

[18] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, “Advances in pre-training distributed word representations,” in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[19] P. Resnik, “Using information content to evaluate semantic similarity in a taxonomy,” in *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 1*, 1995, pp. 448–453.

[20] J. J. Jiang and D. W. Conrath, “Semantic similarity based on corpus statistics and lexical taxonomy,” *arXiv preprint cmp-lg/9709008*, 1997.

[21] C. Leacock and M. Chodorow, “Combining local context and wordnet similarity for word sense identification,” *WordNet: An electronic lexical database*, vol. 49, no. 2, pp. 265–283, 1998.

[22] Z. Wu and M. Palmer, “Verbs semantics and lexical selection,” in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 133–138.

[23] D. Lin, “An information-theoretic definition of similarity,” in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 296–304.

[24] P. V. Panicheva, I. D. Mamaev, and T. A. Litvinova, “Towards automatic conceptual metaphor detection for psychological tasks,” *Information Processing & Management*, vol. 60, no. 2, p. 103191, 2023.

[25] X. He, L. Yu, S. Tian, Q. Yang, J. Long, and B. Wang, “Viemf: Multimodal metaphor detection via visual information enhancement with multimodal fusion,” *Information Processing & Management*, vol. 61, no. 3, p. 103652, 2024.