ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Evaluating Text Representations Techniques for Hypernymy Detection: The Case of Arabic Language

Randah Alharbi and Husni Al-Muhtaseb[*]

Department of Information and Computer Science, [*] IRC for Intelligent Secure Systems
King Fahd University of Petroleum and Minerals (KFUPM)
Dhahran, Saudi Arabia
raharbi@uqu.edu.sa and muhtaseb@kfupm.edu.sa

Abstract. Text representation is a key component in the performance of any hypernymy-related task. In this study, we investigate representation techniques to understand which features best represent the hypernymy relation, focusing on three factors of representation: word embedding, embedding combination techniques, and using features. The results indicate that different embeddings have different effects on performance; concatenation, 'addition and subtraction' have led to better performance, and using unsupervised measures has a negative effect on performance.

1 Introduction

Text representation is a fundamental step in all NLP and IE tasks. Numerous types of word representation exist, from basic sparse and dense representations to complex representations such as neural embeddings. Many tasks have adopted the use of traditional neural word embedding, such GloVe [1]. Contextual embeddings, such as Bidirectional Encoder Representations from Transformers (BERT), provide different representations of a term based on its context [2]. These general word embeddings can model semantic similarity and relatedness between terms that encode various lexico-semantic and topical relations such as synonymy, antonymy, and hypernymy [3]. Some studies have proposed hypernymy-specific representations to better model hypernymy-relation for hypernymy-related tasks. In this work, we experiment with three representation factors for the hypernymy detection task, focusing on Arabic. The objectives of our experimentation are: (1) Evaluate hypernymy-specific embeddings against traditional embedding and contextual embedding. To the best of our knowledge, no other study compares the performance of these types of embedding. (2) Test the effectiveness of vectors' combination techniques for Arabic. (3) Evaluate the effect of adding unsupervised measure values to the representation. We selected several unsupervised measures, Weeds Precision (WeedsPrec) measure [4], Clark Degree of Entailment (ClarkDE) [5], as unidirectional similarity measures, InvCL [6] as directional similarity measure, SLQS_cos [7] as an entropy-based distributional measure and cosine similarity as baseline.

2 Methodology

The main goal of our study is to find the best text representation that models hypernymy relations. We aim to test if hypernymy-specific embedding is better at modeling the hypernymy relation for the hypernymy detection task. To conduct the evaluation experiments, we have selected GloVe embedding as the traditional embedding baseline and BERT as the contextual embedding. For hypernymy-specific embedding, we have selected two retrofitted embeddings, Lexical Entailment Attract-Repel (LEAR) and Generalized Lexical ENtailm-net (GLEN), and two geometrical-based embeddings, Poincare for hierarchical data and Poincare GloVe. To mitigate external effects on the performance, We controlled most of the models' hyperparameters and experimental setups. We also evaluate several mathematical operations for combining terms embedding. We have applied mathematical operations on our baseline embedding and three hypernymy-specific embeddings: LEAR, Poincare GloVe, and GLEN. Finally, we have experimented with enhancing term representations with the values of unsupervised measures as input features. We have used features embedding for one feature and for a combination of two, three, four, and five features. The feature embedding is concatenated with term embeddings before being used as input to the model. In the following subsections, we highlight the details of embedding training corpus, datasets, classification models, and experimental setup.

2.1 Corpus and Datasets

AraBERT corpus: We have trained all word embedding on half the corpus used to train an Arabic version of BERT called AraBERT [8]. It is a collection of Arabic text of 77GB in size and with a vocabulary of 12+ million words. We have used half of the AraBERT corpus for resource utilization training.
Arabic Semantic Relation Dataset (ASRD): We created an in-house dataset for Arabic lexical semantic relationships, extracting data from multiple Arabic semantic resources. It contains one-word examples for hypernym, synonym, meronym, holonym, attribute, antonym, cause, and random words. The number of examples is 246313. ARSD datasets, which will be publicly available.
Evaluation Datasets: We utilized lexical-semantic constraints extracted from the ASRD to train both the embedding models and the classification models. This suggests that having a shared vocabulary might impact the performance of the embeddings. To mitigate this effect, we have selected seven English benchmark datasets containing hypernymy relations from HypEval and translated them into Arabic using Google Translate, namely, BLESS, ENTAILMENT, Lenci/Benotto, Weeds, BIBLESS, and two versions of Root9. We have filtered the terms in these datasets to include only single-word entries that were present in the AraBERT training corpus.

2.2 Representations Training

All representations are trained on half AraBERT corpus except Poincare, which is trained on ASRD hypernymy pairs, and BERT, which is pre-trained on the full AraBERT corpus. For **GloVe** we preprocess the corpus using AraBERT preprocessor and we have used the original GloVe code to train our version. **LEAR** is retrofitting-based embedding that takes pre-trained embeddings as input and modifies the embedding according to lexical-semantic relations constraints. We used GloVe embedding and ASRD constraints as input.LEAR needs

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

synonyms and hypernyms for its Attract objective and antonym for its repel objective. We have used the official Python implementation of LEAR with slight modifications to adapt it to our data and trained for 5, 20, and 100 iterations. **GLEN** also takes pre-trained embeddings and lexical-semantic constraints, but it generates generalized modified embeddings for all vocabulary, even those with no constraints. We have used the official implementation of GLEN with default hyperparameters except for the number of iterations to stop training if there is no improvement. **Poincare GloVe** uses a modified GloVe objective to generate new word embeddings in hyperbolic space. It does not necessarily use pre-trained word embedding or lexical-semantic constraints; rather, we have used the co-occurrence calculation file generated by GloVe training as a basis for its calculation. We have trained two versions **100D Poincare GloVe** trained using 100D Poincare ball and all vocab, and $50 \times 2D$ Poincare GloVe trained in the cartesian product of 50 2D Poincare balls and most frequent words of the vocabulary. **Poincare Embedding** is trained using lexical-semantic constraints with a tree-like structure and we used hypernym and has_instance from ASRD to train it with negative examples set to 5. We have used its Gensim implementation. We have used pre-trained **BERT** for Arabic (AraBERT V2), and for each term, we have extracted features of the final layer output.

2.3 Classification Models and tasks

To Assess the effectiveness of the chosen embeddings in modeling hypernymy relations, we have used the resulting embeddings from each model as input to the hypernymy detection task. The detection model will classify input examples as hypernymy or not, with two classes as output. The positive examples are hypernyms, entailment, and has_instance; other relations are considered negative examples. The goal of our evaluation was not to achieve the highest performance but rather to fairly evaluate representation models by keeping experiment variables consistent among different experiments. Thus, we have used a simple feed-forward neural classification model for each task with an embedding layer, one hidden layer, and an output layer. We have trained a model per embedding. To evaluate the classification models, we test the trained model on several datasets, including the test set of ASRD.

We have evaluated vector combination techniques and the incorporation of unsupervised measures using hypernymy detection models trained on the ASRD dataset and tests on the testing datasets. The representation combination techniques experiments use the same model of hypernymy detection, but we varied the mathematical operations used to combine the pair vectors and the size of the resulting combined embedding. In the unsupervised measures experiments, we have trained models to incorporate every single feature and combination of two, three, four, and five features with term embeddings from GloVe, LEAR5, and 100D Poincare Embeddings. We have created an input features sub-network to enable the learning of feature embeddings. We use cross-entropy loss, Stochastic Gradient Descent (SGD) optimizer, 150 dimensions hidden layer, and ReLU activation function on the output layer and trained for 50 epochs except for models that use BERT representation due to time and computing power limitations.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

3 Results and Discussion

Experiment 1: Evaluating Word representations:
Table 1 shows the result of hypernymy detection using different representations. On ASRD, the best-performing model is Poincare embedding, followed by 100D Poincare GloVe. This is reasonable since Poincare embedding is trained solely on hypernymy examples of ASRD. Moreover, all hypernymy-specific embeddings outperform the GloVe baseline except 50x2D Poincare GloVe. On two datasets, 50x2D Poincare GloVe models outperform others and score similarly to the best embedding on the other two datasets. GLEN outperforms other embeddings on two other datasets and performs similarly to the best embedding model on the three datasets. LEAR5 performs similarly to the best-performing embedding on four datasets. These results suggest that the performance of the hypernymy detection task is highly affected by the datasets. For example, on both BIBLESS and ENTAILMENT datasets, which have the same type of positive and negative examples, the best-performing embeddings are GLEN and LEAR5. Our findings are similar to the findings of [9] for unsupervised hypernymy detection for English, which shows that no unsupervised measure outperforms others on all of their testing datasets because of how the negative samples in a dataset are constructed. Surprisingly, BERT is the least-performing model on the ASRD dataset, which might indicate the difficulty of the hypernymy detection task.

Data set	**GloVe**	**LEAR 5**	**LEAR 20**	**LEAR 100**	**GLEN**	**Poincare Em-bedding**	**100D Poincare GloVe**	**50x2D Poincare GloVe**	**BERT**
ASRD	0.81	0.83	0.82	0.82	0.82	**0.86**	0.83	0.81	0.65
BLESS	0.55	0.55	0.55	0.52	0.52	**0.56**	0.51	0.55	0.41
BIBLE.	0.59	0.65	0.64	0.62	**0.71**	0.51	0.58	0.62	0.63
ENTAI.	0.56	0.59	0.59	0.58	**0.61**	0.51	0.55	0.59	0.53
LB	0.53	0.56	0.56	0.56	0.55	0.45	0.55	0.54	**0.58**
Weeds	0.53	0.54	0.55	0.55	0.55	0.46	0.54	**0.56**	0.53
Root9	0.60	0.61	0.63	0.62	0.59	0.52	0.60	**0.64**	0.56

Table 1: F1-score results for the hypernymy detection task (BIBLE. is BIBLESS, ENTAI. is ENTAILMENT, and LB is LenciBenotto)

Experiment 2: Embedding Combination Techniques:
Table 2 shows the result of arithmetic operations to combine terms' vectors for the hypernymy detection task for GloVe, LEAR5, 100D Poincare GloVe, and GLEN. The results indicate the least effective combination techniques are addition and multiplication. Concatenation, subtraction, 'addition and subtraction', and 'concatenation and subtraction' have similar effects on performance. Meanwhile, concatenation, 'addition and subtraction', and 'concatenation and subtraction' perform slightly better than other operations. Our results for the Arabic language are similar to [10] results for Vietnamese, unlike [3] results, which found that vector difference and concatenation are the best operations for English. This might indicate that vector operations have different effects on different target languages. Nevertheless, in this study, we have used concatenation to preserve all the information of both terms of the relation

Operations	GloVe	LEAR 5	100D Poincare GloVe	GLEN
concat	**0.81**	**0.82**	**0.83**	**0.83**
add	0.72	0.73	0.75	0.77
sub	0.77	0.78	0.82	0.82
mul	0.68	0.63	0.76	0.78
add_sub	0.79	0.80	**0.83**	0.81
concat_sub	0.80	0.79	0.82	0.81

Table 2: F1-score for various embedding techniques and operations.

Experiment 3: Using Unsupervised Measures as Input Features
Unlike our assumption, the results show that no single feature or combination of features is better than the baselines of using no features. Moreover, there is no difference among feature types in their effect on performance. This might be attributed to the new size of the representation, which makes it harder for the model to learn the patterns that indicate hypernymy from the representations. Also, the quality of the learned feature embeddings might be affected. Features embedding might need more epochs to be learned. Moreover, testing on ASRD, the used features might add little information beyond terms embedding. Testing on a different dataset may yield different results, as features could provide additional information about the terms. Table 3 shows the effect of incorporating a combination of four and five features as input.

Features	GloVe	LEAR5	100D Poincare
No features	**0.81**	**0.83**	**0.83**
invCL, clarkeDE, weeds_prec, cosine	0.43	0.78	0.78
invCL, clarkeDE, weeds_prec, SLQS_Cos	0.43	0.78	0.77
invCL, clarkeDE, cosine, SLQS_Cos	0.42	0.78	0.78
invCL, weeds_prec, cosine, SLQS_Cos	0.43	0.78	0.78
clarkeDE, weeds_prec, cosine, SLQS_Cos	0.42	0.77	0.78
invCL, clarkeDE, weeds_prec, cosine, SLQS_Cos	0.43	0.77	0.78

Table 3: F1-score results for using combinations of four and five features as input

3.1 Discussion

From the experiments' results, we observe that, despite being trained without lexical-semantics constraints, 100D Poincare GloVe performs well in the hypernymy detection task. This highlights the effectiveness of modeling the hypernymy relation in hyperbolic space. On the other hand, GLEN outperforms other representations on some of the hypernymy detection datasets, and it falls short with ASRD because GLEN is known to have more impact when used with datasets with fewer known constraints [11]. Generally, the results reveal that no single representation consistently outperforms the others across all evaluation datasets. This suggests that the way training and evaluation datasets are constructed plays a crucial role in performance. This encourages future researchers to develop datasets that incorporate multiple strategies to generate negative hypernymy examples, leading to more robust performance. The results for vector combination techniques on hypernymy detection for Arabic indicate that when computing resources are limited, one might choose the mathematical operation that results in a vector with the least dimensions size (subtraction). Enhancing the representation of terms' embeddings with more information extracted from calculating the unsupervised measures does not enhance the performance of the hypernymy detection task. Further incorporation techniques other than feature embedding should be evaluated.

4 Conclusion

In this work, we investigated the impact of various types of embeddings on the performance of hypernymy detection tasks. Our findings suggest that the choice of dataset used in the training and evaluation has a significant effect on model performance. Moreover, all vector operations perform similarly except addition and multiplication.

Acknowledgment: This work is supported by the Interdisciplinary Research Center for Intelligent Secure Systems IRC-ISS at KFUPM through the grant INSS2405.

References

[1] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[3] Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014.

[4] Julie Weeds and David Weir. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 2003.

[5] Daoud Clarke. Context-theoretic semantics for natural language: an overview. In *Proceedings of the workshop on geometrical models of natural language semantics*, 2009.

[6] Alessandro Lenci and Giulia Benotto. Identifying hypernyms in distributional semantic spaces. In Eneko Agirre, Johan Bos, Mona Diab, Suresh Manandhar, Yuval Marton, and Deniz Yuret, editors, **SEM 2012: The First Joint Conference on Lexical and Computational Semantics*, 2012.

[7] Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, 2014.

[8] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference*, 2020.

[9] Haw-Shiuan Chang, ZiYun Wang, Luke Vilnis, and Andrew McCallum. Distributional inclusion vector embedding for unsupervised hypernymy detection. In *North American Chapter of the Association for Computational Linguistics*, 2017.

[10] Bui Van Tan, Nguyen Phuong Thai, and Nguyen Minh Thuan. Enhancing performance of lexical entailment recognition for vietnamese based on exploiting lexical structure features. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, 2018.

[11] Goran Glavaš and Ivan Vulic. Generalized tuning of distributional word vectors for monolingual and cross-lingual lexical entailment. In *the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.