BL.Research at SemEval-2022 Task 1: Deep networks for Reverse Dictionary using embeddings and LSTM autoencoders

Nihed Bendahman[†], **Julien Breton**[†], **Lina Nicolaieff**[†],
Mokhtar Boumedyen Billami[†], **Christophe Bortolaso**[†], **Youssef Miloudi***[†]
[†]Berger-Levrault, 64 Rue Jean Rostand, 31670 Labège, France
*CARL Berger-Levrault, 361 All. des Noisetiers, 69760 Limonest, France
{nihed.bendahman, julien.breton, lina.nicolaieff, mb.billami,
christophe.bortolaso}@berger-levrault.com, {youssef.miloudi}@carl.eu

Abstract

This paper describes our two deep learning systems that competed at SemEval-2022 Task 1 "CODWOE: Comparing Dictionaries and WOrd Embeddings". We participated in the subtask for the reverse dictionary which consists in generating vectors from glosses. We use sequential models that integrate several neural networks, starting from Embeddings networks until the use of Dense networks, Bidirectional Long Short-Term Memory (BiLSTM) networks and LSTM networks. All glosses have been preprocessed in order to consider the best representation form of the meanings for all words that appears. We achieved very competitive results in reverse dictionary with a second position in English and French languages when using contextualized embeddings, and the same position for English, French and Spanish languages when using char embeddings. Our source code can be found at GitHub[1].

1 Introduction

Distributed representations of words (or word embeddings) (Bengio *et al.*, 2003; Mikolov *et al.*, 2013; Pennington, Socher and Manning, 2014) have shown to provide useful features for various tasks in natural language processing (NLP) and computer vision. While there seems to be a consensus concerning the usefulness of word embeddings and how to learn them, this is not yet clear with regard to representations that carry the meaning of a full sentence. That is, how to capture the relationships among multiple words and phrases in a single vector remains a question to be solved.

Much recent research in computational semantics has focused on learning representations of arbitrary-length phrases and sentences. The reverse dictionary represents one of the most common cases to solve this problem of learning sequence representations. That said, the reverse dictionary is the task to find the proper target word given the word description (Hill *et al.*, 2016; Hedderich *et al.*, 2019; Zhang *et al.*, 2019; Yan, Li and Qiu, 2020). For example, the composed meaning of the words in a dictionary definition (*A mixture of other substances or things*) should correspond to the meaning of the word that define it (*cocktail*). As mentioned by Hill *et al.* (2016), this bridge between lexical and phrasal semantics is useful because high quality vector representations of single words can be used as a target when learning to combine the words into a coherent phrasal representation.

In this paper, we present our contributions to solve the reverse dictionary problem using very specific neural architectures and applying supervised learning. For more information on task 1 of SemEval-2022 as described by its organizers, we invite the reader to consult the paper of Mickus, Timothee *et al.* (2022).

Our approaches require a model able of learning to map between arbitrary-length phrases and fixed-length continuous-valued word vectors. For this purpose, we experiment with two broad classes of neural language models (NLMs): Recurrent Neural Networks (RNNs) with (Bidirectional) Long-Short

[1] https://github.com/jln-brtn/BL.Research-at-SemEval-2022-Task-1

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 94 - 100

Term Memory (BiLSTM and LSTM), which naturally encode the order of input words (or characters), and simpler (feedforward) lexical units embedding models. These lexical units can be bag-of-words (BOW) or a sequence of characters.

After having described in section 2 in more detail the problem to be solved in the SemEval-2022 evaluation campaign and the data provided, we present in section 3 the previous works of reverse dictionary. Then, in section 4, we present our neural architectures with all data preprocessing having been performed. Thereafter, in section 5, we describe the experimental setup implemented before presenting the results in section 6 and concluding in section 7.

2 Background

2.1 Problem Description

The CODWOE shared task[2] consists of compare two types of semantic descriptions: dictionary glosses and word embedding representations. The problem can be defined as follows: given a definition, can we generate the embedding vector of the target word? That said, there are several questions to be solved: (1) How should we compare two very different types of semantic representation? (2) Will contextualized embeddings help better define polysemous (ambiguous) words that have multiple senses? and (3) Can we have the same evaluation performances of the same neural architecture for different natural languages? In other words, five natural languages are studied in this task, namely: English, French, Spanish, Italian and Russian language. Our goal is to answer the following question: can the same model or the same neural network architecture be beneficial for all languages?

2.2 Data Description

The organizing members of the reverse dictionary task proposed different JSON files that contain definitions and their vector representations. Each JSON file describes information about a one natural language for a list of five languages, namely: French, English, Spanish, Italian and Russian language. The corpus is therefore multilingual. Before having the test corpus, the data have been split in different sets: trial, train, and development corpus. All the models we will present have been trained and validated on train and development corpus.

Before describing the data, we can do a quick focus on the size of these data. As we mentioned, the data are split into 3 groups. For each language, the trial dataset contains 200 elements pairs (definitions and their embedding vectors), the train dataset contains 43,608 elements pairs, and the development dataset contains 6,375 elements pairs. Regarding the test corpus, we have 6,208 definitions for each language. The organizers have provided different vector representations for the definitions. All these representations are continuous vectors (embedding vectors). They have 256 dimensions and are built with well-known three techniques:

- "char" corresponds to character-based embeddings, computed using an auto-encoder on the spelling of a word.
- "sgns" corresponds to Skip-Gram with negative sampling embeddings, aka. Word2Vec (Mikolov *et al.*, 2013).
- "electra" corresponds to Transformer-based contextualized embeddings.

As mentioned above, all datasets of SemEval-2022 task 1 are multilingual. This is an important point because we can imagine and create a system that manages these multilingual datasets, or several systems specialized in one language. We will explore this option in the “Experimental Setup”.

3 Related Work

In the field of natural language processing, word embeddings have been the subject of several research problems for many years. Indeed, a text contains various information, and the idea is to resorb the target information in a continuous vector representation. Embedding’s methods improve significatively the results against standard statistical approaches and justified the interest these last years. In order to create embeddings, the scientific research community is experimenting with two approach types: unsupervised and supervised learning of embeddings.

The unsupervised approaches are the most common and consist in using a pretrained language model on a large corpus such as Word2Vec (Mikolov *et al.*, 2013) or Glove (Pennington, Socher and Manning, 2014). From the word

[2] https://competitions.codalab.org/competitions/34022

embeddings we can obtained, we must then choose a technique in order to find the right combination of words that will best convey the desired information. For example, it's possible to apply combinations such as: average, sum, or centroid to obtain a vector that reflect the representation for a sequence of words. We can find an example of centroid usage with Lwin and Nwet (2019) for news summarization extraction or a centroid vector weighted by IDF (Inverse Document Frequency) (Arora, Liang and Ma, 2017).

For the supervised approaches to produce sequence text vector representation, the idea is consisting of modeling the link between a content and an embedding representation. The reverse dictionary is a common case of sequence or short content representation. Some state-of-the-art models are used to perform this task, for example neural networks LSTM (Sherstinsky, 2018) and Bert (Pre-training of Deep Bidirectional Transformers for Language Understanding) (Devlin *et al.*, 2018). Notably in the paper of Yan, Li and Qiu (2020), they experiment with word generation from a definition using Bert multilingual architecture. As it's mentioned in their paper, the use of a Bert model is a great idea and can, at least, achieves state-of-the-art performances for both monolingual and cross-lingual reverse dictionary task. Even better, the proposed framework here can perform cross-lingual reverse dictionary task without aligned data.

We can also talk about the work of Morinaga and Yamaguchi (2020), Malekzadeh, Gheibi and Mohades (2021) which are based on a Long Short-Term Memory (LSTM) architecture. With always our objective to produce a vector of a contextualized text, the LSTM offers great prospect in this field of research. Indeed, the recurrent neural network architecture of the LSTM allow models to perform on sequential data which is exactly our case study in this task.

4 System Overview

In this section, we describe the models we proposed in the CODWOE – Reverse dictionary shared task. In order to keep comparable and linguistically significant the results submitted by the different participants, the organizers of CODWOE disallowed any use of external resources, including standard datasets as well as pretrained models that could be used for this task (such as Word2Vec models or contextual pretrained models based on Transformer's architectures like Bert). Given this condition, we decided to explore the sequential models, and particularly the LSTM and BiLSTM models. All the following models' architecture we created are based on TensorFlow[3] and Keras[4] (Chollet and others, 2015) libraries.

4.1 Data preprocessing

Before introducing our models, we want to mention that we have performed preprocessing on the content data. By using Stanza[5] (Qi *et al.*, 2020), we lemmatized all definitions and removed all punctuations. We decided to do this to minimize alternative words for the same concept and help our models to correctly process the vocabulary.

To optimize our workflow, we worked on the data before and independently from the neural network architectures. In this way, we built new files based on the lemmatization of the main corpus. This process is possible because all languages are covered by Stanza[6].

4.2 Baseline Model

In this section, we will introduce our first model, called: Baseline Model. This model is intentionally simple in order to create baseline scores and introduce manipulation on the datasets. The figure 1 presents our first architecture.

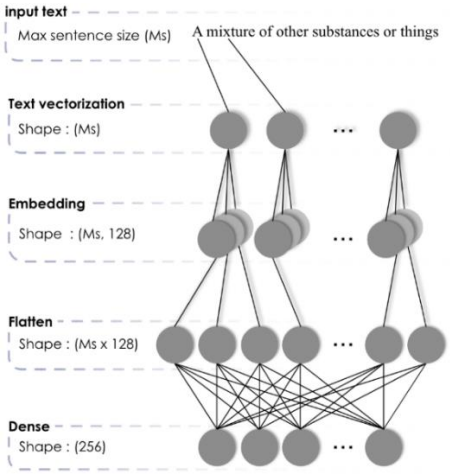

Figure 1 : Baseline Model Architecture.

[3] https://www.tensorflow.org/
[4] https://keras.io/
[5] https://stanfordnlp.github.io/stanza/
[6] https://stanfordnlp.github.io/stanza/available_models.html

As you can see on the figure 1, our model created with Keras contains four different layers and starts with the text vectorization. This first layer will transform the input text in a vector to be process by the next layers. To perform this operation, we must give an identifier to each different word in our corpus. After that, each sentence will be represented as a vector of identifiers. To be processed, the vector of a sentence must have the same size for all sentences. That said, we take the maximum sentence size (Ms in figure 1) and normalize all the vectors by adding zero values in the end.

Now, each vector's sentence is ready to be processed by the embedding layer. This layer turns positive integers (indexes) into dense vectors of fixed size. After this operation, the flatten layer will change the dimensionality of the data from two dimensions to one dimension without losing any value. The shape of this layer will be the multiplication of the two dimensions of the previous layer. Finally, we model the output data by using a fully connected layer (Dense layer) with 256 dimensions to match with the gloss embeddings gave by the organizers.

4.3 Advanced Model

LSTMs are a Recurrent Neural Networks (RNN) (Medsker and Jain, 2001) which have an internal memory that allows them to store the information learned during training. LSTMs are frequently used in the reverse dictionary task (Sherstinsky, 2018) and in word and sentence embeddings tasks in general (Augustyniak, Kajdanowicz and Kazienko, 2019; Liu *et al.*, 2020), as they can learn long-term dependencies between existing words in the sentence and thus compute context representation vectors for each word. BiLSTM for its part, is a variant of LSTMs, it allows a bidirectional representation of words (Augustyniak, Kajdanowicz and Kazienko, 2019).

Our second model, named Advanced Model, is therefore a BiLSTM-LSTM network. As for the baseline model, we use a sequential model which can be provided by Keras. The figure 2 presents our second architecture. This last one starts with a text vectorization layer, followed by an Embedding layer and a dense layer producing vectors of the words passed as an input, the vectors have (length of the longest sentence, 128) dimensions. Then we added a BiLSTM layer, which takes a recurrent layer (the first LSTM of our network) which in turn takes the "merge mode" as an argument. This mode specifies how the forward and reverse outputs should be combined; in our case the average of the outputs is taken.

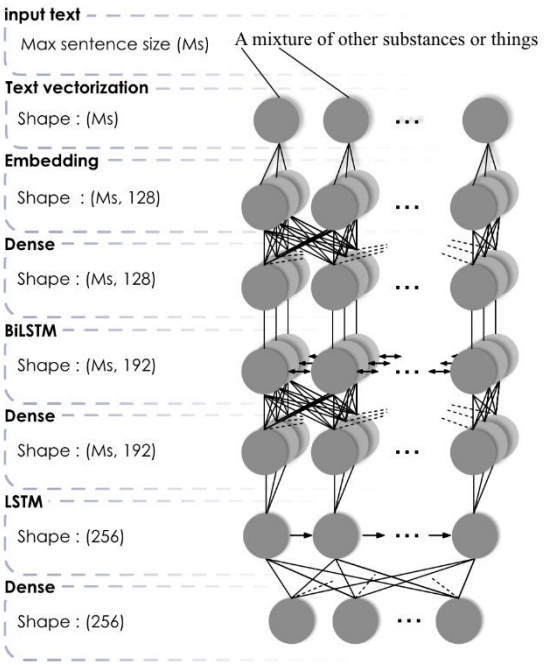

Figure 2 : Advanced Model Architecture.

To these three layers, we added another fully connected Dense layer and a LSTM layer of 256 dimensions corresponding to the dimensions of the output vectors and a final Dense layer with the same dimensions as illustrated in figure 2. We use the Softmax as an activation function. For the hyper-parameters to train the model, we use the following: epochs = 10; batch size = 192; learning rate = 1e-3 and AdamW as an optimizer.

5 Experimental Setup

In this section, we describe different variants we tested. Since there were 3 types of vector representations proposed to us in this shared task, we used the same architectures to produce the 3 types of vectors. However, the data format given as input to the model is not the same for the 3 types. For the 'electra' and 'sgns' representation types, we prepare a vocabulary containing the words of the glosses of the 'training dataset', the words of this vocabulary were obtained by following the preprocessing described in section 4.1.

For the 'char' vector type, we construct a vocabulary of all the characters used in the glosses without preprocessing the data. The idea being that,

for the 'char' type representation, the model encodes the characters of the glosses into vectors and then produce the vectors encoding the glosses based on the vectors of the characters constituting the glosses.

The model that we propose is a monolingual model, i.e., we trained it separately on the training dataset of each language provided. However, in order to evaluate the impact of using a multilingual model, we trained the same neural networks on five languages (with character vector) at the same time and compared the results obtained with those obtained by the monolingual models.

For the 'sgns' and 'electra' representation types, we built a vocabulary containing the words of all glosses on the five languages, which contains in total 121,147 words. We did the same with the 'char' vectors but with preparing a vocabulary of characters instead containing 405 characters, in total. The table 1 describes the vocabulary size for each monolingual model and for multilingual model.

Model Type	Language	Vocab Type	
		Words	Chars
Monolingual Model	English	21,001	139
	French	24,089	170
	Spanish	**29,383**	**229**
	Italian	25,414	162
	Russian	29,289	212
Multilingual Model	All languages	**121,147**	**405**

Table 1: Vocabulary size of models.

We can see that there are common words between the different languages since the multilingual model has a vocabulary of 121,147 words instead of 129,176. That said, there are 8,029 common words between at least two language vocabularies. Moreover, we find that the vocabulary of the Spanish language is the best represented in the train dataset.

6 Results and Analysis

In this section, we present the performance results on using architectures that we described in section 4 and try to give clue to understand them. Our main goal was to outperform the organizers' baseline model and results (Mickus, Timothee *et al.*, 2022).

For our first model (our baseline), the model is not better than the state-of-the-art models for this task. However, we can analyze an interesting point: this simple model surprisingly produces better results on the rank cosine (Rank) measure. To illustrate this remark, we can look the model results in table 2. On results for the MSE measure, only 3 cases outperform the organizers' baseline model. Moreover, every rank cosine measure is better. At this point, we can reach our first analyze, it's hard to perform in the MSE and Cosine (Cos) with, at the same time, trying to obtain good results in Rank (and vice-versa). This analyze is supported by the following table 3 based on advanced model.

Language	Emb-type	MSE	Cos	Rank
EN	char	**0.216**	0.709	0.449
	electra	1.638	**0.805**	0.433
	sgns	1.217	0.165	0.311
FR	char	0.501	0.690	0.428
	electra	1.394	**0.813**	0.441
	sgns	1.867	0.166	0.314
ES	char	0.632	0.787	0.411
	sgns	1.089	0.251	**0.253**
IT	char	0.691	0.572	0.417
	sgns	1.329	0.245	**0.246**
RU	char	**0.165**	0.787	0.409
	electra	0.946	0.694	0.398
	sgns	0.690	0.219	0.289

Table 2: Our Baseline model results.

Language	Emb-type	MSE	Cos	Rank
EN	char	**0.143**	0.795	0.500
	electra	1.326	**0.843**	0.500
	sgns	0.895	0.153	0.500
FR	char	0.365	0.769	0.500
	electra	1.112	**0.857**	0.500
	sgns	1.106	0.211	0.500
ES	char	0.510	0.824	0.500
	sgns	0.910	0.227	0.500
IT	char	0.358	0.728	0.500
	sgns	1.111	0.227	0.500
RU	char	**0.132**	0.829	0.500
	electra	0.864	0.719	0.500
	sgns	0.566	0.298	0.425

Table 3: Advanced model results.

With our second model, the results are completely opposite. We performed in MSE and Cosine measure. With these two measures, we're

doing better than the organizers' baseline model. On the other hand, the Rank cosine seems to be stuck on 0.5. We can also compare our results with the other participants. Our BiLSTM-LSTM architecture is efficient on 'char' and 'electra' embeddings. For example, in 'char' with English, French and Spanish languages, we obtain the second-best score over the seven participants in SemEval-2022 campaign at task 1. We can conclude this analyze for the advanced architecture with this open-ended question: Why our architecture performs on English, French and Spanish but seam to give worse results on the Italian and Russian languages?

As we mentioned earlier, we tried to create a multilingual model. Unfortunately, after trained this model on all five languages and test on French Character embeddings, the model gave us poor results: 0.67 for MSE and 0.48 for Cosine measure. These are the worst results we've had in this competition, so we decided to drop this architecture and focus on the models presented in the system overview section.

Given the set of results obtained, we find that the best cosine score was obtained by using electra (contextualized) vector embeddings and the best MSE score was obtained by using character vector embeddings. More generally, the use of BiLSTM-LSTM architecture neural network has been beneficial in having results that surpass baselines when cosine and MSE are used as evaluation measures.

7 Conclusion

In this paper, we have presented our contributions to solve the task 1 problem of the semeval-2022 evaluation campaign. We studied the effects of training sentence embeddings with supervised data by testing on five different languages, namely: English, French, Spanish, Italian and Russian language. We showed that models learned with char embeddings or contextualized embeddings can perform better than models learned with Skip-Gram word embeddings. By exploring various architectures, we showed that the combination of Embedding/Dense/BiLSTM/ Dense/LSTM layers can be beneficial than the simple use of Embedding layer.

We believe that the neural architecture of our advanced model can be used to solve other tasks such as Definition Modeling (Noraset *et al.*, 2017), where the objective would be to reverse the inputs/outputs of the model, or other natural language processing tasks where the objective is to add a specific output layer to adopt the specific problem like sequence classification, for example.

Acknowledgments

We would like to thank Berger-Levrault for giving us the opportunity to participate in the 2022 edition of SemEval. Our thanks also go to the research and technological innovation department (DRIT) which is an important part of Berger-Levrault.

References

Arora, S., Liang, Y. and Ma, T. (2017) 'A Simple but Tough-to-Beat Baseline for Sentence Embeddings', in *International Conference on Learning Representations (ICLR)*.

Augustyniak, L., Kajdanowicz, T. and Kazienko, P. (2019) 'Aspect Detection using Word and Char Embeddings with (Bi)LSTM and CRF', *CoRR*, abs/1909.01276. Available at: http://arxiv.org/abs/1909.01276.

Bengio, Y. et al. (2003) 'A Neural Probabilistic Language Model', *J. Mach. Learn. Res.*, 3, pp. 1137–1155.

Chollet, F. and others (2015) Keras. GitHub. Available at: https://github.com/fchollet/keras.

Devlin, J. et al. (2018) 'Bert: Pre-training of deep bidirectional transformers for language understanding', arXiv preprint arXiv:1810.04805 [Preprint].

Hedderich, M.A. et al. (2019) 'Using Multi-Sense Vector Embeddings for Reverse Dictionaries', arXiv:1904.01451 [cs] [Preprint]. Available at: http://arxiv.org/abs/1904.01451 (Accessed: 17 February 2022).

Hill, F. et al. (2016) 'Learning to Understand Phrases by Embedding the Dictionary', *Transactions of the Association for Computational Linguistics*, 4, pp. 17–30. doi:10.1162/tacl_a_00080.

Liu, Y. et al. (2020) 'Depth-Adaptive Graph Recurrent Network for Text Classification', CoRR, abs/2003.00166. Available at: https://arxiv.org/abs/2003.00166.

Lwin, S.S. and Nwet, K.T. (2019) 'Extractive Myanmar News Summarization Using Centroid Based Word Embedding', in *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 200–205. doi:10.1109/AITC.2019.8921386.

Malekzadeh, A., Gheibi, A. and Mohades, A. (2021) ‘PREDICT: Persian Reverse Dictionary’, arXiv:2105.00309 [cs] [Preprint]. Available at: http://arxiv.org/abs/2105.00309 (Accessed: 24 February 2022).

Medsker, L.R. and Jain, L.C. (2001) ‘Recurrent neural networks’, *Design and Applications*, 5, pp. 64–67.

Mickus, T. et al. (2022) ‘SemEval-2022 Task 1: CODWOE -- COmparing Dictionaries and WOrd Embeddings’, in *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

Mikolov, T. et al. (2013) ‘Efficient estimation of word representations in vector space’, arXiv preprint arXiv:1301.3781 [Preprint].

Morinaga, Y. and Yamaguchi, K. (2020) ‘Improvement of Neural Reverse Dictionary by Using Cascade Forward Neural Network’, *Journal of Information Processing*, 28(0), pp. 715–723. doi:10.2197/ipsjjip.28.715.

Noraset, T. et al. (2017) ‘Definition Modeling: Learning to Define Word Embeddings in Natural Language’, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI Press (AAAI’17)*, pp. 3259–3266.

Pennington, J., Socher, R. and Manning, C.D. (2014) ‘Glove: Global Vectors for Word Representation.’, in *EMNLP*, pp. 1532–1543.

Qi, P. et al. (2020) ‘Stanza: A Python Natural Language Processing Toolkit for Many Human Languages’, CoRR, abs/2003.07082. Available at: https://arxiv.org/abs/2003.07082.

Sherstinsky, A. (2018) ‘Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network’, CoRR, abs/1808.03314. Available at: http://arxiv.org/abs/1808.03314.

Yan, H., Li, X. and Qiu, X. (2020) ‘BERT for Monolingual and Cross-Lingual Reverse Dictionary’, arXiv:2009.14790 [cs] [Preprint]. Available at: http://arxiv.org/abs/2009.14790 (Accessed: 17 February 2022).

Zhang, L. et al. (2019) ‘Multi-channel Reverse Dictionary Model’, arXiv:1912.08441 [cs] [Preprint]. Available at: http://arxiv.org/abs/1912.08441 (Accessed: 17 February 2022).