TLDR at SemEval-2022 Task 1: Using Transformers to Learn Dictionaries and Representations

V. Harsha Vardhan[†]
IIIT Hyderabad
harshavardhan.v
@research.iiit.ac.in

Aditya Srivastava[†]
IIIT Hyderabad
aditya.srivastava
@research.iiit.ac.in

Abstract

We propose a pair of deep learning models, which employ unsupervised pretraining, attention mechanisms and contrastive learning for representation learning from dictionary definitions, and definition modeling from such representations. Our systems, the **T**ransformers for **L**earning **D**ictionaries and **R**epresentations (**TLDR**), were submitted to the *SemEval 2022 Task 1: Comparing Dictionaries and Word Embeddings (CODWOE)*, where they officially ranked **first on the definition modeling subtask**, and achieved competitive performance on the reverse dictionary subtask. In this paper we describe our methodology and analyse our system design hypotheses.

1 Introduction

Dictionaries are some of the linguistically richest resources available for a language, in addition to being extremely clean and unbiased in comparison to most naturally occurring language data, which is noisy and shows domain specific bias based on its source. Thus, there has been considerable interest towards using NLP models to harness this knowledge, especially for low-resource languages. Broadly there are two sets of approaches towards the same - the first is to use dictionaries for representation learning and using these representations for transfer learning in other tasks, such as in the work by Bosc and Vincent (2018) where they use an LSTM based auto-encoder to learn rich representations from dictionary definitions such that the definitions can also be generated back from the representations. Tissier et al. (2017) also used dictionary definitions to build sets of 'strong' and 'weak' pairs of words to get improved word representations with greater interpretability by moving words which show a stronger semantic-relatedness closer together in the embedding space.

The second approach has been to move in the opposite direction, such as in the work by Chang and Chen (2019) where the authors try to map contextualized word representations to their dictionary definitions in an effort towards word-sense disambiguation. This has been further explored by Noraset et al. (2016), where they use an RNN based language model to generate definitions for representations. Recent work by Bevilacqua et al. (2020) improves upon this by leveraging pre-trained encoder-decoder models like BART (Lewis et al., 2019) in order to generate the definitions of words. These pre-trained language models, significantly outperform the RNN based models. The ability to generate definitions for representations makes these contextual representations of words explainable.

NLP has advanced leaps and bounds within the past decade, with a major push coming from the advent and utilization of transfer learning via representation learning techniques such as Word2Vec (Mikolov et al., 2013a) and ELMO (Peters et al., 2018). The more recent methods to employ transfer learning use large pretrained language models, such as BERT (Devlin et al., 2019) and XLM (Conneau et al., 2020). These models are jointly used with unsupervised training objectives such as MLM and Causal-LM, to transform natural language into information rich meaning representations which are then used for many different downstream tasks. We aim to replicate the same in our experiments to give a better prior for the model, before it learns to generate desired representations.

One of the characteristics common to all the language models mentioned above is that they are all based on transformers. Transformers use the multi-headed attention and self-attention mechanisms to learn extremely effective language representations. Transformers also scale well since unlike their predecessors, the RNNs, they process information in parallel and thus are much faster.

For SemEval 2022's Task 1, Comparing Dictio-

[†] *Authors contribute equally to this work.*

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 60 - 67

naries and Word Embeddings (Mickus et al., 2022), the participants were asked to design systems for the following two subtasks;

1. Subtask 1: Reconstruct SGNS (Mikolov et al., 2013b), character and ELECTRA (Clark et al., 2020) embeddings from their dictionary glosses.

2. Subtask 2: Reconstruct the dictionary glosses from their SGNS, character and ELECTRA embeddings.

The subtasks are called the Reverse Dictionary and Definition Modeling subtasks respectively. The first subtask is evaluated on the Mean Squared Error (MSE), Cosine Similarity and Cosine Ranking between the generated representations and the gold representations. The second subtask is evaluated on Mover score, Sense level BLEU score (S-BLEU) and Lemma level BLEU score (L-BLEU).

In this system description paper we detail our model architectures, training, evaluation and testing methodologies, and try to analyse our hypotheses and their impact on the final scores. For the reverse dictionary subtask we designed a simple BERT-like model, pretrained it on the MLM objective, and finetuned it for the subtask on a combination of cosine embedding loss and MSE loss, alongside negative sampling of the dataset to add a contrastive loss to the overall objective function. For the definition modeling subtask we designed a model based on transformer decoders for natural language generation, with masked self-attention over the inputs in addition to multi-head attention over the representations from the three embeddings spaces. We submitted our systems for evaluation over English data and our systems demonstrated very good performance in the contest itself, with the definition model system outperforming all other submissions and taking first place, and the the reverse dictionary system achieving the fifth, sixth and seventh place on the character, ELECTRA and SGNS targets respectively. Lastly, we also show some post-contest improvements on the reverse dictionary system. The code for our experiments has been open sourced and is available on GitHub.[1]

2 Subtask 1: Reverse Dictionary

2.1 Data Preprocessing

We maintain the data splits provided by the task organizers (43608 training samples, 6375 dev samples and 6221 test samples) with each sample containing the dictionary gloss and its SGNS, character and ELECTRA representations. All models were trained on the training split, with the best model picked from evaluation over the dev split.

The dictionary glosses were lower-cased and stripped of all whitespace characters except those essential to maintaining word boundaries for tokenization. We also padded and truncated all sequences to a maximum sequence length of 256 tokens.

For contrastive learning we performed negative sampling to augment each gloss-embedding pair with three other embeddings from the same semantic space and the same data split.

2.2 System Overview

We designed our system around three hypotheses - firstly, using pretraining would work better than starting from scratch since it would give the model a good prior for the downstream tasks (transfer learning). Secondly, training individual models for each representation would outperform training a single multitask model, as the representations do not reside in a common semantic space. Thirdly, optimizing over a combination of losses for each of the metrics that we're being evaluated upon, i.e. MSE loss for MSE, Cosine Embedding (CE) loss for cosine similarity, contrastive loss for cosine-rank, would work better than using any of them individually.

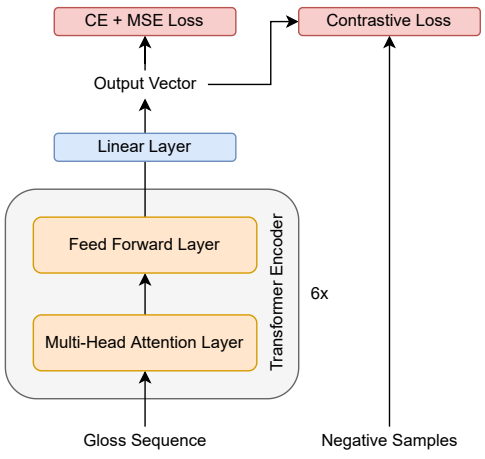

Figure 1: System architecture for the Reverse Dictionary subtask.

The foundation of our representation learner is based on the DistilBert (Sanh et al., 2019) architecture and comprises of a stack of 6 transformer

[1] https://www.github.com/IamAdiSri/tldr-semeval22

encoders, with 12 attention heads each, hidden dimension of 3072 and embedding dimension of 768.

We start by pretraining the model via unsupervised masked-language-modeling (Devlin et al., 2018) over only the texts from the dictionary glosses in the task dataset. Individual instances of this pretrained model are then appended with a linear layer of dimension 256 to project outputs in the required dimensions, and fine-tuned for each semantic space, optimized over the following loss function;

$$\begin{aligned} L = & e^{-\log(p_0)} * \mathit{MSE}(\phi(g), v_p) + \\ & e^{-\log(p_1)} * \mathit{CE}(\phi(g), v_p) + \\ & e^{-\log(p_2)} * \mathit{CL}(\phi(g), v_{n0}, v_{n1}, v_{n2}) \end{aligned} \tag{1}$$

where, $\phi(g)$ is the sentence embedding for a gloss g, v_p is the true (or *positive*) embedding, v_{n0}, v_{n1} and v_{n2} are the negative samples and p_0, p_1 and p_2 are trainable parameters for weighting the different loss functions. **MSE** and **CE** are the **Mean Squared Error** and **Cosine Embedding Loss** respectively, which function as the reconstruction loss between the generated representation and true representation. Lastly, **CL** is the **Contrastive Loss** between the generated representations and the false representations from negative sampling. The equations for the three are given below;

$$\mathit{MSE}(a, b) = \frac{1}{d}\sum_{i=0}^{d}(a_i - b_i)^2 \tag{2}$$

$$\mathit{CE}(a, b) = 1 - \frac{a.b}{|a||b|} \tag{3}$$

$$\mathit{CL}(p, n_0, n_1, n_2) = \sum_{i=0}^{2} 1 - \mathit{CE}(p, n_i) \tag{4}$$

where a, b, p and n are all vectors of size d.

2.3 Experimental Setup

We used Pytorch (Paszke et al., 2019) and the HuggingFace (Wolf et al., 2019) library to write our experiments in Python. Though the HuggingFace library does provide ready-to-go pretrained language models, they were not used in our experiments or submissions. We did however use a ready-made pretrained tokenizer from the library to tokenize our texts.[2] This was permitted by the organizers and we believed would be a significant advantage over training a new tokenizer from scratch, considering that the dataset is rather small and homogeneous in its linguistic variation. All models were trained on Nvidia's GTX-1080 Ti and RTX-2080 Ti GPUs.

We pretrained our model on the unsupervised MLM objective, with a learning rate (LR) of $5e-5$, masking probability of 0.15, and an effective batch size of 64 samples. For fine-tuning we use an LR of $5e-3$ with a linear LR scheduler and an effective batch size of 64 samples per batch. Any remaining hyperparameters were left as their default values. We selected the language model that attained the lowest perplexity score (55.25) over the dev split.

For models that were multitask trained, instead of a single output layer (as shown in figure 1) we have three output layers (one for projecting into each semantic space), all receiving the same output vector from the transformer encoders below them.

While finetuning we train all models for 50 epochs in total, and select a model for each semantic space and metric based on the epoch that attains the best score over the dev split. The same model is used for producing results over the test split.

2.4 Results and Analysis

The experiments here were designed with the intent to evaluate the three modeling hypotheses outlined at the beginning of section 2.2, with tests on pretraining, mutitasking and the objective function. The results for these experiments are given in tables 1, 3 and 4.

Repr.	Metric	Pretrained	RndInit.
sgns	mse	0.8990793	**0.8987827**
sgns	cos	**0.1805207**	0.1799421
sgns	rnk	**0.5004269**	0.5004292
char	mse	0.1465276	**0.1454727**
char	cos	0.7897581	**0.7916019**
char	rnk	**0.5004282**	0.5004290
electra	mse	1.5150244	**1.3510013**
electra	cos	**0.8452746**	0.8455241
electra	rnk	**0.5000801**	0.5000807

Table 1: Comparison between using pretraining versus starting from a randomly initialized model.

As we can see from table 1, there is no significant difference (mostly under $1e-4$) between

[2] https://huggingface.co/distilbert-base-uncased

Repr.	MSE	CSim.	Rank	PreTr.	MltTsk.	Objs.
sgns	558.96838	**0.18531**	0.50043			CE
sgns	**0.89953**	0.17866	**0.50043**	✓	✓	MSE, CE, CL
char.	125.12206	**0.79565**	0.50043			CE
char.	**0.14337**	0.79560	**0.50043**	✓	✓	MSE, CE, CL
electra	49.98565	**0.84564**	0.50008			CE
electra	**1.34014**	0.84563	**0.50008**	✓	✓	MSE, CE, CL

Table 2: Best submissions on the Reverse Dictionary leaderboard.

finetuning a pretrained model versus training the model from scratch. We did however notice that our pretrained models seemed to converge in fewer epochs than when the models were trained from scratch, indicating that the pretraining did have some positive effect.

Repr.	Metric	Individual	Multitask
sgns	mse	**0.8984921**	0.8990793
sgns	cos	0.1786035	**0.1805207**
sgns	rnk	0.5004290	**0.5004269**
char	mse	**0.1431219**	0.1465276
char	cos	**0.7955332**	0.7897581
char	rnk	0.5004292	**0.5004282**
electra	mse	**1.3292400**	1.5150244
electra	cos	0.8451633	**0.8452746**
electra	rnk	**0.5000672**	0.5000801

Table 3: Comparison between individual models for each semantic space versus a single multitask model.

Table 3 displays the results of our tests comparing a single multitask model for all three semantic spaces versus training individual models for each. Again the results are quite inconclusive whether one reliably outperforms the other, however considering differences of over $1e-3$ to be significant (as they affect the leaderboard rankings) we can see that individual models outperform multitask models on a greater number of metrics.

Finally, from the ablation tests in table 4 we can clearly see that while optimizing over the combination of losses or MSE alone gives comparable scores, optimizing over CE loss alone causes the MSE score to worsen manyfold. This seems to imply that MSE contributes significantly more than CE and CL losses to the performance of the models. The scores also demonstrate that tuning over MSE tends to improve the MSE metric, while tuning over CE improves the cosine similarity, agreeing with our hypothesis about the same. Contrary to our expectations however, we can also observe that adding contrastive learning by negative sampling does not improve the cosine-ranking by much.

3 Subtask 2: Definition Modeling

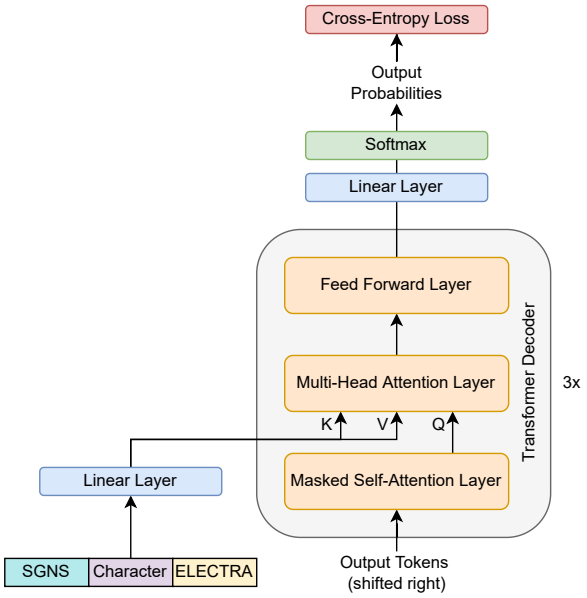

Figure 2: System architecture for the definition modeling subtask.

3.1 Data Preprocessing

The dictionary glosses were first normalized for punctuation and then tokenized using the Moses scripts (Koehn et al., 2007). We then learn a subword tokenizer on the training data, in order to create a fixed vocabulary of subwords. The Moses tokenized sentences were used to learn Byte Pair Encodings (BPE) with a vocabulary size of 10,000 using the subword-nmt library (Sennrich et al., 2016).

Repr.	Metric	All Losses	Only MSE	Only CE
sgns	mse	0.8991734	**0.8956623**	694685.50
sgns	cos	0.1787637	0.1547712	**0.1852663**
sgns	rnk	**0.5004290**	**0.5004290**	0.5004290
char	mse	**0.1430358**	0.6462155	169896.47
char	cos	0.7955229	0.3045915	**0.7955511**
char	rnk	**0.5004290**	0.5004294	**0.5004290**
electra	mse	**1.3285819**	3.5256984	82728.45
electra	cos	0.8451307	0.0650083	**0.8453690**
electra	rnk	0.5000807	**0.5000807**	0.5000808

Table 4: Ablation tests on the reverse dictionary objective function.

3.2 System Overview

The definition modeling problem is posed as one of text generation, generating the definition D_* of a word w_* autoregressively, given the semantic representation of the word in the form a vector v_*. Therefore, we maximise the likelihood of the definition $D_* = \{w_0, w_1, .., w_n\}$, where w_i corresponds to the ith word of the definition, given the vector v_*.

$$P(D_*/v_*) = \prod_{i=0}^{n} P(w_i/w_0, ..w_{i-1}, v_*) \quad (5)$$

The definition modeling architecture that has been used in this system is a transformer decoder, whose output softmax layer approximates the above likelihood.

In a vanilla transformer-seq2seq architecture (Vaswani et al., 2017), the decoder in the self-attention layer projects the decoder states into three matrices called Query (Q), Key (K) and Value (V) and using the below equation, attention values are computed.

$$\textit{Attn.}(Q, K, V) = \textit{Softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \cdot V \quad (6)$$

This is followed by a cross-attention layer where the decoder attends to the encoder states by projecting the encoder states as the Keys, Values and using the decoder states as the Queries.

In our model, since the input vectors are not in the same space as the decoder embeddings, a linear layer is used to learn a projection between them. The output of this layer is then projected into the Key (K) and Value (V) matrices which are used along with the decoder self-attention outputs to compute the cross attention values.

$$f = \textit{linear}(v_{\textit{sgns}} \oplus v_{\textit{char}} \oplus v_{\textit{electra}}) \quad (7)$$

where v_{sgns}, v_{char}, $v_{electra}$ represent the SGNS, character, ELECTRA vectors respectively and $\oplus$ denotes concatenation.

$$\begin{aligned} K &= f \cdot W^K \\ V &= f \cdot W^V \\ Q &= o \cdot W^Q \end{aligned} \quad (8)$$

where W^K, W^V are the matrices learnt to project the output of linear layer (f) to Key, Value matrices and W^Q is the matrix to project the output of the decoder self-attention layer o to the Query matrix.

3.3 Experimental Setup

The experiments for the defmod subtask were carried out using the Fairseq framework (Ott et al., 2019) in Python. All the models were trained on Nvidia GTX 1080 Ti GPUs.

The transformer decoder model consists of 3 decoder layers, with each layers consisting of 8 attention heads and an embedding dimension of 512 with the input and the output embeddings of the decoder being tied. The model is trained using label smoothed cross entropy loss with label smoothing of 0.2. A learning rate of 5e-4 using an inverse square root scheduler with a weight decay of 0.0001 was used to optimize over a batch size of 4096 tokens.

The model was trained for over 50 epochs and the checkpoint with the least validation perplexity

Word	Gloss	Predicted Gloss
farming	A farming operation ; a farm , or instance of farming on a piece of land .	The act or process of producing food
hazard	The chance of suffering harm ; danger , peril , risk of loss .	Something that causes trouble or destruction
hardware	Equipment .	Of or relating to a computer system
transition	The process of change from one form , state , style or place to another .	The act or process of converting
hastily	In a hasty manner ; quickly or hurriedly	In a hurried manner
chesty	Not dry ; involving the coughing of phlegm .	Of, pertaining to, or characteristic of a dove
Solarian	Of or relating to the Solar System .	Of or pertaining to the planet Mars.
valid	Well grounded or justifiable , pertinent .	Not able to be true.
alt-left	The extreme or radical left of the political spectrum .	A member of the United States of the United States of the United States of the United States of the United States.

Table 5: Examples of generated glosses.

was selected to generate definitions for the test set using beam search with a beam size of 10.

3.4 Results and Analysis

The ablation study results in table 6 shows the impact of each vector, namely SGNS, char and ELECTRA, against the best performing model where features were extracted using a concatenation of all three.

Repr.	Mover Score	S-BLEU	L-BLEU
All	**0.12847**	**0.03278**	**0.04250**
Electra	0.11008	0.02957	0.03629
Char	0.10403	0.02884	0.03643
SGNS	0.03622	0.01743	0.02114

Table 6: DefMod scores using all vs individual representations

We can clearly see that the ELECTRA representations outperform char and SGNS, with the SGNS vectors falling significantly behind on all three metrics. It can also be observed that by concatenating all three representations and extracting useful features using attention significantly boosts performance. This allows one to infer that char and SGNS vectors do contain semantic information that ELECTRA does not.

The submission utilizing all representations outperformed all other submissions in the defmod subtask and ranked first for English.

3.4.1 Observations Post Dataset Release

After the passing of the task deadline, the organizers released the full dataset, complete with all annotations. With this data we were able to make further observations about our model by comparing the generated glosses to the actual glosses in the dataset. A few examples are shown in table 5. Examples marked in red indicate wrong or irrelevant definitions, and the ones marked in green describe the relevant ones.

From the generated glosses in green, we can see that the model shows a remarkably good mapping between the words and their representations, and makes close approximations of their meanings when generating glosses. In the example of *farming*, we can see that the model is able to correctly associate the act of farming with that of producing food. The words *hazard* and *transition* are also correctly associated with *trouble (danger)* and *conversion (change)* respectively, demonstrating that the model is able to recall similar concepts. The model shows good language proficiency as well, with syntactically correct utterances.

In case of words like *hardware*, which have multiple meanings, we can see that the model outputs one of the secondary definitions that it has learnt

from the dataset. From this, we can infer that in the absence of appropriate context (i.e. how the word is being used in a sentence), the model has difficulty in disambiguating the word's meaning, although it still recognizes the word and picks one of the correct definitions.

Finally, the model tends to learn definition templates from the training data, such as *"Of, pertaining to"* or *"Of or relating"*, that it reuses during generation. As a result of being a sequence to sequence model, it also occasionally exhibits degenerate repetition, as seen in the *"alt-left"* example.

4 Conclusion

In this paper, we explored a variety of approaches towards dictionary definitions and embeddings generation, especially under the constraint of being unable to use external monolingual data. We have analyzed the effectiveness of each of these methods, performing ablation studies showing the impact that various objective functions like MSE, cosine embedding loss and contrastive loss have in reconstructing representations from text, and coming up with an attention mechanism utilizing all the provided representations to generate definitions to produce exceptional results.

As part of future work, we plan to explore the performance of our models in multilingual settings. We would also use test the performance of these models against pre-trained language models like BART, BERT etc. to gauge the impact language model pre-trainig since we did not have enough monolingual data in the tasks to train them. Finally, we plan to experiment with a joint training mechanism where instead of training on each subtask independently, the models can inform and improve each other by collaboratively learning both the subtasks.

References

Michele Bevilacqua, Marco Maru, and Roberto Navigli. 2020. Generationary or "how we went beyond word sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7207–7221, Online. Association for Computational Linguistics.

Tom Bosc and Pascal Vincent. 2018. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, Brussels, Belgium. Association for Computational Linguistics.

Ting-Yun Chang and Yun-Nung Chen. 2019. What does this word mean? explaining contextualized embeddings with natural language definition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6064–6070, Hong Kong, China. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, page 177–180, USA. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Timothee Mickus, Denis Paperno, Mathieu Constant, and Kees van Deemter. 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.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2016. Definition modeling: Learning to define word embeddings in natural language.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Copenhagen, Denmark. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.