Evaluating BERT-based Approaches for Reverse Dictionaries

Alex Xie, Arvin Wu, Jinqi Chen, Kai Franz*
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{alexx, arvinw, jinqic, kfranz}@andrew.cmu.edu

Abstract

The reverse dictionary task is that of predicting a word given its definition or natural language description. Reverse dictionaries have a number of applications, from aiding in second language acquisition to helping writers better express their ideas. In this project, we replicate the current state-of-the-art BERT-based reverse dictionary models, extending them using variants of BERT and employing a variety of different training objectives. We employ models based on the SentenceBERT and CharacterBERT variants of BERT, and we introduce a novel BERT-based model utilizing multilabel classification that outperforms both baselines on unseen data.

1 Introduction

Given the definition or description of a word, can we accurately find that word? This is the reverse dictionary problem, one that has a wide range of applications ranging from helping writers better express themselves to aiding in second language acquisition to solving the "tip of the tongue" problem. The past few years have witnessed great interest in reverse dictionaries, with approaches ranging from database-driven models to recurrent neural networks.

Nevertheless, recent work still leaves much to be desired, as current commercial-use reverse dictionaries like **OneLook.com** may perform very poorly on certain queries, while also lacking generalization abilities. For example, for the definition "a native of a cold country," OneLook returns words like "foreign", "indigenous", "native" while missing more appropriate words like "eskimo". The newest BERT-based models such as WantWords, the current state of the art in reverse dictionaries, ameliorate such issues.

Hence, we plan to pursue two main goals in our project: (1) confirm and replicate the results of two current state-of-the-art models, WantWords [17] and BERT [16], and (2) improve upon the baselines by designing and training new architectures and the variants of current state-of-the-art models.

2 Related Work

2.1 Feature-based Approach

In the past few years, feature-based approaches have been popular towards the reverse dictionary problem. Several feature-based approaches have been proposed, which rely much heavily on embeddings. The earliest approaches use two broad classes of neural language models (NLMs), namely Recurrent

*Alphabetical order

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Neural Networks (RNNs) and bag-of-words (BOW) embedding models, and use Word2Vec embeddings to map between arbitrary-length phrases and fixed-length continuous-valued word vectors [4]. A node-graph structure has also been proposed in order to establish the semantic similarity between words and definitions [14]. In their approach, they assume that the significance of the meaning of a word to a definition is proportional to its frequency throughout the definition in the forward dictionary (FD). They implement a graph-search through related words (relation through definition), and use a distance-based similarity measure to compute the target words. *Definition modeling* is also introduced for generating a definition for a given word and its embedding, and can be used to rank a set of test words based on how likely they are to correspond to a given definition [10].

Later, more variants of the feature-based models are proposed and achieve better results on the reverse dictionary task. The cascade-forward neural network (CFNN) with biLSTM and noising data augmentation are proven to outperform the state-of-the-art commercial OneLook Reverse Dictionary [9]. The node-graph structure is also extended with a Multi-Sense LSTM (MS-LSTM), which achieves promising results on the reverse dictionary task [5]. Simple sense-level distinction, like coarse-grained *supersenses*, are shown to lead to more accurate semantic understanding and significant performance improvements for the reverse dictionary task [11].

Recently, the Multi-channel Reverse Dictionary Model has achieved state-of-the-art performance for feature-based models [17]. In their model, the confidence score of a word prediction is a combination of the confidence scores from both internal channels (POS tag predictor and morpheme predictor) and external channels (word category predictor and sememe predictor), and is trained with a biLSTM with attention sentence encoder. The POS tag allows the model to distinguish between words with similar definitions but different parts of speech; the category predictor uses WordNet lexical names to differentiate between related words that belong to unrelated categories. These additional channels mitigate the problems caused by highly variable input queries and low-frequency target words.

2.2 Transformer-based Approach

In recent years, transformer-based approaches have also been popular in the reverse dictionary domain. A masked BERT model, which predicts the words corresponding to "[MASK]" tokens inserted into the input, converts the prediction for "[MASK]"s into the word ranking list [16]. It produces promising results for both previously unseen words and for human-generated word descriptions.

WantWords, an open-source online reverse dictionary system, replaces the biLSTM with attention in the Multi-channel Reverse Dictionary Model with a (pre-trained) BERT model to encode the input definition or description [12]. Specifically, the hidden state corresponding to the initial $\langle$CLS$\rangle$ token is used as a "sentence embedding". The encoded sentence vector produced is then passed through separate single-layer perceptrons to yield a word score, a part of speech score, and a category score. In addition, the hidden states from the BERT encoder are collectively used to obtain morpheme and sememe scores. These five components are then combined to produce final predictions for each word. WantWords has significant dominance in performance over existing baseline models.

3 Methods

3.1 BERT

BERT has, in recent years, achieved state-of-the-art results in a number of language modelling tasks [3]. In this section, we will give a brief overview and background of the basic BERT architecture before presenting in later sections various methods of applying pre-trained BERT models and their variants to the reverse dictionary task.

BERT consists of the the encoder portion of a transformer [15], which itself is composed of l encoder layers, each of which can be described as follows (note that we use the BERT base model, in which $l = 12$):

$$\begin{aligned} h_{sub}^{(l)} &= \text{LayerNorm}(h_{l-1} + \text{MultiHeadAttention}(h_{l-1})) \\ h^{(l)} &= \text{LayerNorm}(h_{sub}^{(l)} + \text{FFN}(h_{sub}^{(l)})) \end{aligned} \tag{1}$$

where the feed-forward network (FFN) is a two-layer MLP with dropout applied to each position of the input sequence (i.e. a 1-D convolution of width 1), and layer normalization is a transform that normalizes neuron inputs [1].

Further, multi-head attention can be expressed as

$$\begin{aligned} &\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h])W^O \\ &\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2}$$

where Q, K, V are the attention queries, keys, and values, W_i^Q, W_i^K, W_i^V, W^O are learned projection matrices (note that the former three are unique to each head), h is the number of heads (a hyperparameter), Concat denotes concatenation, and where each (scaled dot product) attention is given by

$$\text{Attention}(Q, K, V) = \text{Softmax}\Big(\frac{QK}{\sqrt{d_k}}\Big)V \tag{3}$$

where d_k is the dimension of the keys and queries (which is independent of the dimension of the values). As its output, BERT generates for each token in the input sequence a corresponding contextual embedding vector.

The process of training BERT for any given problem consists of two stages: pre-training, in which the model learns to generally model language, and fine-tuning, in which the model's knowledge is transferred to a specific problem (in this case, reverse dictionary). BERT is pre-trained on two tasks, masked language modelling (MLM) and next sentence prediction (NSP), the former of which is of particular interest to us. Masked language modelling entails predicting the identity of certain masked, or hidden, tokens in a sentence - for example, given the sequence "I put ⟨MASK⟩ on my hot dog", the model should predict "ketchup" or "mustard" as the masked token. Meanwhile, in the next sentence prediction task, given one sentence followed by another, the goal is to determine whether the second sentence can logical follow the first (i.e. output yes/no).

Note also that BERT uses BPE tokenization, a greedy and deterministic tokenization algorithm, to encode its input, contrasting with the word- or character-level approaches of other models. Under this tokenization scheme, out-of-vocabulary words may be broken up into a sequence of several sub-word tokens - for instance, "loanee" is tokenized as ["loan", "#ee"]. Furthermore, BERT uses several special tokens, most notably ⟨CLS⟩, which serves as a start-of-sequence marker and whose corresponding output vector is used in the NSP task, ⟨MASK⟩, which is used to mask tokens in the MLM task, and ⟨MASK⟩, which is used both as a separator between sentences in the NSP task and as an end-of-sequence marker.

In the next section, we describe a baseline model that reframes the MLM training objective for the reverse dictionary task.

3.2 BERT for Reverse Dictionary (BERT-MLM)

Yan et al., 2020 [16] describes a procedure for fine-tuning a pre-trained BERT model for the reverse dictionary task using masked language modelling (MLM). For each training instance consisting of some target word and its definition, the model is provided as input the sequence

$$s = \Big[\langle\text{CLS}\rangle, \underbrace{\langle\text{MASK}\rangle, ..., \langle\text{MASK}\rangle}_{\langle\text{MASK}\rangle \text{ token repeated } m \text{ times}}, \langle\text{SEP}\rangle, d^{(1)}, ..., d^{(|d|)}, \langle\text{SEP}\rangle\Big]$$

where $d = [d^{(1)}, d^{(2)}, ..., d^{(|d|)}]$ is the BPE-tokenized definition sentence and m, the number of mask tokens injected into the sequence, is a hyperparameter.

We can then pass s through BERT, obtaining contextual embeddings for all the tokens in the sequence, and take solely the embeddings corresponding to the m mask tokens and pass them through a masked language model head, a 2-layer MLP. The MLM head outputs for each mask token a distribution over all sub-word tokens. Formally,

$$\begin{aligned} P_{\text{subword}} &= \big[P_{\text{subword}}^{(1)}, ..., P_{\text{subword}}^{(m)}\big] \\ P_{\text{subword}}^{(i)} &= \text{MLM}(H_{\text{mask}}^{(i)}) \end{aligned} \tag{4}$$

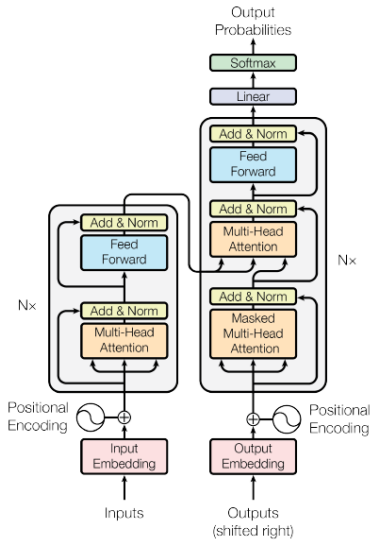

Figure 1: The transformer architecture presented in (Vaswani et al., 2017). BERT uses the encoder portion of the transformer, shown on the left half of the diagram, and omits the decoder, on the right half of the diagram.

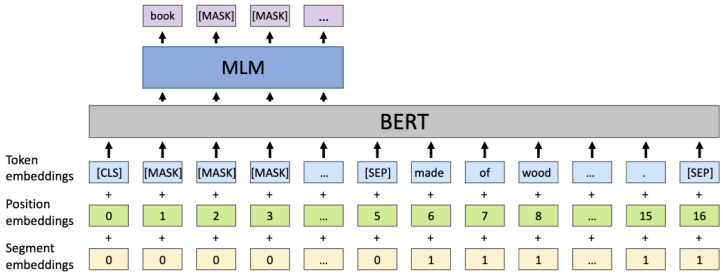

Figure 2: Model architecture and masked training objective presented in [16].

where H_{mask} denotes the sequence of m contextual embeddings corresponding to the m mask tokens in the input s. Note that $P_{\text{subword}} \in \mathbb{R}^{m \times S}$, where S is the number of subword tokens used by BERT.

We can convert this sequence of probabilities of subwords to a probability distribution over vocabulary words as follows. We first obtain the BPE tokenization w of every word in the vocabulary, then pad it to length m if it consists of fewer than m tokens or truncate it to length m if it contains more.

$$w' = \begin{cases} w & |w| = m \\ \left[w^{(1)}, ..., w^{m}\right] & |w| > m \\ \left[w^{(1)}, ..., w^{(|w|)}, \underbrace{\langle\text{MASK}\rangle, ..., \langle\text{MASK}\rangle}_{\text{repeated } m-|w| \text{ times}}\right] & |w| < m \end{cases}$$

Then, to calculate the (log) probability of any given word, we can sum the probabilities that each masked token is filled by the token at the same position in the tokenized and padded/truncated word sequence. Thus, for any word w, we can compute:

$$P_{\text{word}}(w) = \sum_{i=1}^{m} P^{(i)}_{\text{subword}}(w'^{(i)}) \tag{5}$$

where $P^{(i)}_{\text{subword}}(w'^{(i)})$ denotes the probability of the i-th mask being filled with token $w'^{(i)}$. Note that $P_{\text{word}}(w)$ is a scalar value.

Finally, having obtained this distribution over all words, we can compute the cross-entropy loss to maximize the probability of the target word:

$$\text{Loss} = -\log\left(\frac{\exp(P_{\text{word}}(w_{\text{target}}))}{\sum_{w\in V}\exp(P_{\text{word}}(w))}\right) \tag{6}$$

where w_{target} is the target word and V is the reverse dictionary vocabulary.

3.3 Extensions to BERT for Reverse Dictionary (BERT-ML-MLM)

In practice, users of a reverse dictionary rarely have a single word in mind. For instance, if a user provides the description "the opposite of happiness," it is likely that they will accept not just "sadness", but also synonyms like "sorrow", "gloominess", or "depression." Hence, the training objective of BERT for Reverse Dictionary, in which only one word is targeted for any given definition, is somewhat unrealistic.

To ameliorate this asymmetry between training and practical usage, we convert the problem from a (single label) classification task to a multi-label (ML) classification task. This is done by leveraging WordNet, an English lexical database [8]. WordNet structures lexical data in the form of synsets, or collections of lemmas (the WordNet equivalent of words) with similar meanings, i.e. **synonyms**. These synsets are linked according to a hypernym-hyponym semantic relation, where **hypernyms** are more general or broad in scope, while **hyponyms** are more granular. For instance, "eagle" is a hyponym of "bird" (in other words, an eagle is a type of bird), while "bird" is a hypernym of "eagle." Further, each lemma is associated with a collection of **derivationally-related** forms, lemmas which are derived from some root lemma. For instance, "learned" is derived from "learn".

For any word w, let $\text{Syn}(w)$, $\text{Hypo}(w)$, $\text{Hyper}(w)$, and $\text{Deriv}(w)$ denote the WordNet synonyms, hyponyms, hypernyms, and derivationally-related forms of w, respectively. Our training objective then, is to map the definition d of w to all $w' \in T(w)$, where

$$T(w) = \{w\} \cup \text{Syn}(w) \cup \text{Hypo}(w) \cup \text{Hyper}(w) \cup \text{Deriv}(w)$$

Thus, treating each word in the vocabulary as an independent label, we can use a binary cross entropy criterion to compute our loss.

$$\begin{aligned}
\text{Loss} &= -\sum_{v\in V}(y_v \cdot \log x_v + (1-y_v)\cdot\log(1-x_v)) \\
y_v &= \begin{cases} 1 & v \in T(w) \\ 0 & v \notin T(w) \end{cases} \\
x_v &= \text{Sigmoid}(P_{\text{word}}(v))
\end{aligned} \tag{7}$$

where V is the reverse dictionary vocabulary.

However, in order to ensure that our model still learns the relationship between the definition and the specific target word, we scale the binary cross entropy of the target word by a factor t. Furthermore, due to the extreme sparsity of "positive" examples for each word in our vocabulary, we introduce a positive weight p, scaling the loss for positive instances (i.e. words we want to map our definition to). As a result, our updated loss function is as follows.

$$\begin{aligned}
\text{Loss} &= -\sum_{v\in V} s_v \cdot (p \cdot y_v \cdot \log x_v + (1-y_v)\cdot\log(1-x_v)) \\
y_v &= \begin{cases} 1 & v \in T(w) \\ 0 & v \notin T(w) \end{cases} \\
s_v &= \begin{cases} t & v = w \\ 1 & v \neq w \end{cases} \\
x_v &= \text{Sigmoid}(P_{\text{word}}(v))
\end{aligned} \tag{8}$$

3.4 SentenceBert for Reverse Dictionary

SentenceBERT [13] is an alternative method to obtain embedding of a given phrase. During the pre-training stage, a Siamese network consisting of two transformer encoder models (i.e. the BERT

architecture) with tied weights is used. Given a pair of sentences, each sentence will be fed into a BERT transformer, followed by a pooling layer [13]. Finally, their output embeddings, as well as the element-wise difference of the two embeddings, will be fed into a linear layer. Ultimately, the result of this pre-training is a model that outputs "semantically meaningful sentence embeddings that can be compared using cosine similarity."

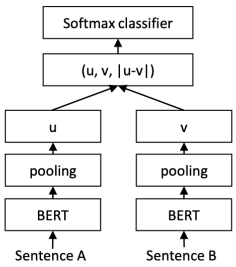

Figure 3: SentenceBert's pre-training architecture

We hypothesize that using these embeddings of word definitions rather than the contextual embeddings outputted by BERT may improve performance on the reverse dictionary task. We place an MLP based loosely on the BERT masked language modelling head on top of the SentenceBERT model, passing the computed sentence embedding through this MLP to obtain a probability distribution over all words in our reverse dictionary vocabulary. We can then perform fine-tuning using cross entropy.

$$S_{\text{embed}} = \text{SentenceBERT}(d) \tag{9}$$

$$P_{\text{word}} = \text{MLP}(S_{\text{embed}}) \tag{10}$$

where d is the BPE-tokenized definition sentence. Note that d does not have any masked tokens, and that our SentenceBERT model does not employ the masked language modelling objective used by our BERT model. This is because the sole semantically meaningful output of SentenceBERT is the sentence embedding; there are no such guarantees for the contextual embedding it produces for a single mask token.

3.5 CharacterBert for Reverse Dictionary

CharacterBERT, introduced by Zheng et al. in 2020 [2], improves the state-of-the-art performance of BERT by producing robust, word-level, and open-vocabulary representations. CharacterBERT is similar in every way to the vanilla BERT, but it uses a different method to construct the initial context-independent representations: instead of splitting the input phrases into multiple wordpieces then embed each unit independently using a wordpiece embedding matrix, it uses a Character-CNN module which consults the sequence of characters to produce a single representation.

For example,

$$\begin{aligned} \text{Apple} &\longrightarrow \text{Ap \#\#ple} \quad \text{(BERT)} \\ &\longrightarrow \text{A p p l e} \quad \text{(CharacterBERT)} \end{aligned}$$

Before diving into our architecture, we would like to give a brief overview of Character-CNN. Character-CNN is implemented as part of ELMo's architecture, where each word is converted to a sequence of characters with a maximum sequence length of 50. The character sequence is then fed into multiple 1-d CNNs with filters $[1, 32], [2, 32], [3, 64], [4, 128], [5, 256], [6, 512], [7, 1024]$. The CNN representation then goes through two Highway layers that apply non-linearities with residual connections before being projected to a final embedding of 768 dimensions.

In our model, we are inspired by the CharacterBERT model as well as the BERT with masked language modelling. We use a [MASK] token in the first index in order to capture all information in a definition into the output contextual embedding. Specifically, given a definition d,

$$s = \left[\langle\text{CLS}\rangle, \langle\text{MASK}\rangle, \langle\text{SEP}\rangle, d^{(1)}, ..., d^{(|d|)}, \langle\text{SEP}\rangle\right]$$

At the end of the CharacterBERT model, we extract the output embedding for [MASK], and use a MLP layer with weight initialization to compute a softmax over the total number of vocabularies in the dictionary. Namely,

$$C_{\text{embed}} = \text{CharacterBERT}(d) \tag{11}$$

$$P_{\text{word}} = \text{MLP}(C_{\text{embed}}[1]) \tag{12}$$

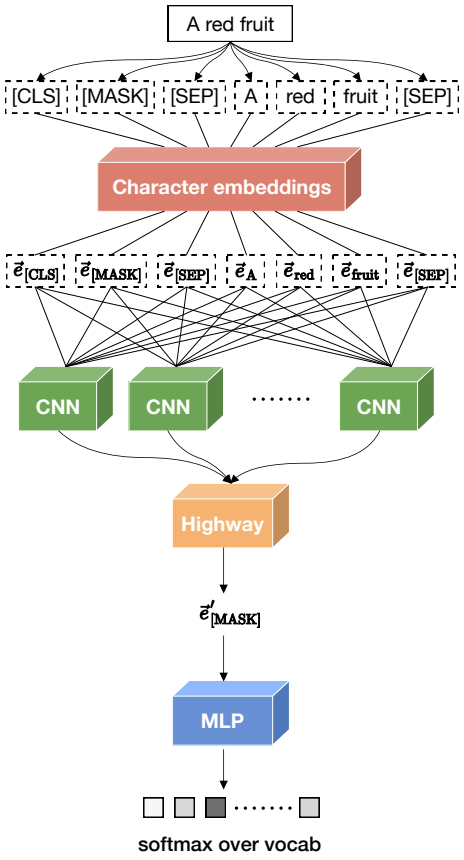

Figure 4: Character BERT model with mask.

3.6 Embedding-based BERT Approach (BERT-w2v)

The reverse dictionary can be framed as a regression problem as well (as was done by [4]) by utilizing word2vec embeddings [7]. Specifically, we employ a BERT + MLP model which maps the contextual embedding of the [CLS] token to the vector space of word2vec (we use the 300-dimensional word2vec variant). We thus fine-tune by minimizing the cosine similarity between the outputs of the model and the word2vec embeddings, which forces the outputs to approximate the word2vec embeddings in high-dimensional space. In inference, we rank the words in terms of proximity of their word2vec embeddings to the embedding computed by our model.

3.7 Translation Model

The reverse dictionary can be framed as a translation task, where the target language is the set of words and the source language is the set of definitions. Using the Transformer architecture introduced by Vaswani et al. in [15], we trained a translation model to learn this task, although it performed significantly worse than the baseline models.

4 Experimental Evaluation

4.1 Dataset

In accordance with past work, we are using the WordNet-based dataset of word-definition pairs constructed by Hill et al. 2016 [4]. Cumulatively, this dataset consists of approximately 750,000 instances, split into training, validation, and testing sets as shown in Table 1. As a measurement of generalization ability, the held-out validation set consists solely of words not included in the training set.

	Training	Validation	Seen Test	Unseen Test	Desc. Test	Total
Size	676k	76k	500	500	200	753k
Unique Words	45k	5k	500	500	200	50k

Table 1: Breakdown of our reverse dictionary dataset

We measure our models' performance on three distinct test sets: a **seen** set consisting of 500 previously seen words, an **unseen** set consisting of 500 unseen words and their definitions, and a **description** set consisting of 200 human-created descriptions of seen words. These three sets each test different aspects of our model. The first, the seen set, measures the **recall**, or capacity, of our model. Given that it contains the same data we are training on, we expect the performance of our model on this set to increase monotonically as it trains. The second, the unseen set, tests the **generalizability** of our model. It is the most challenging of the three sets as it requires the model to extend its knowledge to predict words it has never seen before. Finally, the description set measures the **robustness** of our model to noisy or inaccurate inputs, as the human-generated descriptions are not as precise as the dictionary definitions the model is trained on. We report our models' performances on all three test sets in Table 2.

4.2 Performance Evaluation Metrics

We evaluate our models using the **1/10/100** metric introduced by Hill et al., 2016 [4]. This metric measures the proportion of test examples in which the target word appears among the top 1, 10, 100 most likely words predicted by the model. Additionally, we take the **median rank** of the target word in the model prediction list (lower better, minimum is zero) as well as the **variance** of the rank of the target word (lower better). We choose these metrics for two main reasons: 1) they allow us to easily compare our models to past work, the vast majority of which uses these metrics and 2) they provide insight at a sufficient level of granularity into the performance of our model.

4.3 Description of Baseline

We choose two state-of-the-art transformer-based models, **WantWords** [12] and **BERT-MLM** [16], (WantWords described in Related Work, BERT-MLM described in 3.2) as our baselines. These differ mainly in the fact that WantWords employs a wealth of feature-engineering in addition to BERT, while BERT-MLM relies solely on the linguistic information latent in BERT. We chose these two as our baselines as they have the best performance in the reverse dictionary literature. We reimplemented the models as faithfully as possible, referring to their open source code for hyperparameters and specific implementation details, such as data cleaning techniques.

4.4 Experiments

For our experiments, we leverage open-source pre-trained transformer models. We use the `bert-base-uncased` tokenizer from Huggingface Transformers[2] as the tokenizer for our BERT and SentenceBERT models (no tokenizer is required for CharacterBERT). In addition, we use the the `bert-base-uncased` model from Huggingface Transformers[2], the `general_character_bert` model obtained by the authors of [2][3], and the `distilbert-base-nli-stsb-mean-tokens`

[2] `https://github.com/huggingface/transformers`

[3] `https://github.com/helboukkouri/character-bert`

model from SentenceTransformers[4] as encoders for our BERT, CBERT and SBERT models, respectively.

For our BERT-ML-MLM model, we use the majority of the same hyperparameters used in our BERT-MLM baseline [16], as well as the same pre-trained masked language modelling head from Huggingface Transformers[5]. However, though we train for twenty rather than ten epochs due to the added difficulty of the training objective. Due to time and resource constraints, we were unable to try many configurations of our two additional hyperparameters of our loss function, the target weight t and the positive weight p. Ultimately, among the configurations we tried, $(t, p) \in \{(5, 2), (5, 10), (25, 2), (25, 10)\}$, we found $t = 25$ and $p = 10$ to yield best results, which we report below.

For all our decoder MLPs, we use two fully connected linear layers, one mapping the dimension of BERT encoding (768) to the dimension of BERT encoding (768), and another mapping the dimension of BERT encoding (768) to the vocabulary size for a softmax distribution of possible outputs, with GELU activation and LayerNorm between the two layers. In order to make the inputs suitable for the `general_character_bert` model specifically, we added tokens '[CLS]', '[MASK]', '[SEP]' at the front of each sentence and '[SEP]' at the end of each sentence, and then feed the encoder output for '[MASK]' into the decoder.

When training the two BERT variants, SBERT and CBERT, we trained a frozen version of BERT (disallowing the backpropagation of gradients) as well as an unfrozen version of BERT for each, selecting the highest scores we can achieve with both. For SBERT, we used a learning rate of 6e-5 with Adam optimization, and used linear schedule with warmup (warmup duration 0.01). For CBERT, we used a learning rate of 2e-5 for the encoder (in the unfrozen case) and a learning rate of 1e-3 for the decoder with SGD optimization. This is because after training SBERT, we suspect that having higher learning rate for the decoder layers could improve the performance on unseen data, but it turns out that CBERT's performance on unseen data is almost the same as SBERT despite we tune the learning rate. We used a batch size of 32 for both models. We also used cross entropy loss and weight initialization for both models.

	Seen			**Unseen**			**Description**		
	rank	1/10/100	var	rank	1/10/100	var	rank	1/10/100	var
OneLook	0	**.66/.94/.95**	200	-	-	-	5.5	.33/.54/.76	332
WantWords	16	.19/.43/.73	299	49	.09/.29/.59	351	4	.31/.65/.88	219
BERT-MLM[6]	0	.61/.88/.93	228	68	.08/.31/.53	434	1	**.42/.75/.94**	98
BERT (w2v)	34	.14/.39/.61	387	211	.03/.19/.39	439	38	.09/.33/.62	369
BERT-ML-MLM	1	.36/.86/.95	191	16	**.11/.43/.70**	359	3	.28/.73/.93	70
SBERT (frozen)	1	.55/.81/.86	340	1000	0/.01/.01	89*	4	.28/.63/.85	242
SBERT (unfrozen)	3	.33/.64/.80	350	1000	0/.01/.01	83*	1	.38/.70/.85	245
CBERT (frozen)	1	.39/.67/.81	343	1000	0/0/.01	82	10	.24/.50/.70	356
CBERT (unfrozen)	0	.51/.79/.86	338	1000	0/.01/.01	89	5	.29/.60/.86	218

Table 2: Reverse dictionary model performance. The first row is the results from ([16]). The second and third row are our replicated results of the baselines. The fourth to ninth rows are our experimental results of BERT variant models.

5 Results

We find that across all models and all metrics, BERT with the masked language modelling (MLM) objective performs strongest. Specifically, we find that our **BERT-ML-MLM** (multilabel masked

[4] `https://github.com/UKPLab/sentence-transformers`

[5] `https://huggingface.co/transformers/model_doc/bert.html#bertformaskedlm`

[6] our re-implementation of [16]

language model) variant yields particularly strong results on the unseen dataset, beating the baseline models in median rank and accuracy. This indicates that injecting additional lexical information into our model in the form of related WordNet words is beneficial to reverse dictionary performance, at least with regard to ability to generalize to unseen words. This may be due to the fact that unseen words may show up as synonyms or other related forms of seen words, enabling the model to predict them by association. This also belies the importance of the quality of the lexical database used to extract such information, as incorrect or sparse lexical data may have reduced the performance of our model.

The generally strong performance of masked BERT models speaks to the strength of the masked language model training objective, which aligns with the BERT pre-training objective, allowing for effective transfer learning. Our results on the seen data still lag behind those of OneLook, though this is a somewhat unfair comparison as OneLook is not a neural model but a database that indexes numerous dictionaries, which likely contain the exact definition being input. Nevertheless, our BERT MLM models provide very strong results on the seen dataset as well.

The BERT (w2v) model does not perform as well as the baseline models. We suggest that this is because word2vec embeddings do not take into account any contextual information, whereas BERT model generates contextual embeddings for the input definition. By forcing the outputs of the BERT + MLP model to suit into the word2vec embedding space, much contextual information is lost, resulting in less optimal performance.

Our CharacterBERT and SentenceBERT models provide passable to strong results on seen and description data, even beating WantWords and approaching BERT-MLM in certain respects, but are unable to generalize whatsoever to unseen data. The most likely cause of this is overfitting in the output MLPs of both models. In particular, we suspect that since the unseen words never appear in the training data, the MLPs learn to minimize loss by always setting their probabilities to zero. This explanation agrees with Figure 5 - from the unseen histogram, we can see that almost all target words in the unseen set are never assigned post-softmax scores greater than near-zero by our SentenceBERT model, implying that even when their definitions are given, the model stubbornly refuses to predict them.

If this is indeed the issue, it may be ameliorated by introducing our multilabel training objective into these models as well, enabling them to learn unseen words just as our BERT-ML-MLM model did. In addition, note that this issue does not afflict our BERT-MLM and BERT-ML-MLM models because they use a pre-trained masked language modelling head MLP, which does not need to learn from scratch the relationship between BERT contextual embeddings and the subtokens. Note also that we could not use the pre-trained MLM MLP for our CharacterBERT and SentenceBERT models because they output distributions over the target word vocabulary, not over subword tokens.

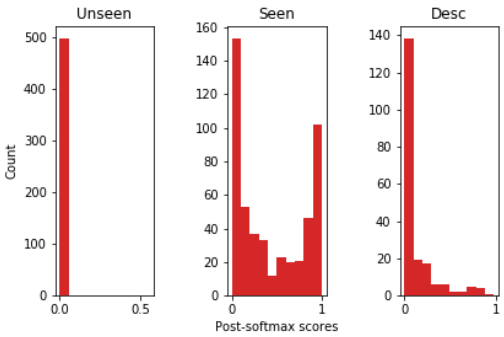

Figure 5: For each of the unseen, seen, and description test sets, we track for each target word in the set the maximum post-softmax score assigned by our SentenceBERT model over all examples in the set. The distribution of scores is shown in the histogram above.

6 Conclusion

6.1 Contributions

Over the course of the project, we tackled the problem of reverse dictionary on a pre-processed dataset using multiple deep learning models. Our contributions can be summarized as:

1. Replicating and confirming the state-of-the-art models Wantwords and BERT-MLM, which are our two baseline models.
2. Exploring different kinds of architectures by modifying and extending the existing models using variants of BERT.
3. Studying the effects of masking, multilabeling, SentenceBERT, CharacterBERT, as well as the neural translation model and how well it can suit for a variety of different training objectives.
4. Proving that the transformer-based approach works and beats the baseline models.
5. Laying the foundations to extend current architectures for reverse dictionary and give insights into future work.

6.2 Discussion

For the reverse dictionary task, there are two main approaches: feature-based and transformer-based approaches. Based on our results, we found that engineering the features such as morpheme, sememe, word category etc. seems to generate better results on the seen dataset, as shown in OneLook, and transformers such as BERT provides huge benefits for generalization in the unseen and description dataset, as shown in BERT-MLM and BERT-ML-MLM. It seems hard (even for the current state-of-the-art model) to achieve high performance for all of the seen, unseen, and description datasets.

One interesting observation (as shown in 6 and 7) is that nearly all models, including the baseline models and our models, have varied outputs regarding similar definitions such as "a very intelligent person" and "a very smart person". The output words for "a very intelligent person" oriented more towards "brain", while the output words for "a very smart person" geared more towards "smart" itself. This is not desired, as we would like more stability for slightly varying definitions. We propose that a better way to tackle this problem is to augment the dataset by randomly replacing the words in the definitions by their synonyms.

Another interesting qualitative result is that in certain cases, our models that yield poorer quantitative results may in fact yield better predictions. For instance, in Table 7, we see that among our neural models, the only one that manages to yield the gold standard word “genius” is SentenceBERT, despite the fact that it does not perform strongest on the description set. Furthermore, in Table 7, where the majority of our models fail to yield “feudalism” (a member of our unseen dataset), the BERT-w2v model manages to generate it. This implies that an ensemble of our models may yield stronger results than any one alone, a potential topic for future study.

6.3 Future Work

In the future, we aim to beat the state-of-the-art model by proposing a new model that has promising performance on all of seen, unseen, and description test data. We can achieve this by incorporating the advantages of both feature-based approach and transformer-based approach. We would also experiment with the different values of positive or target weight for the BERT-ML-MLM model in order to further improve its performance. We would augment our dataset by randomly replacing words in the definitions by their synonyms in order to generate more stable results.

In addition, it may help to incorporate word sense data into our model. Currently, in our BERT-ML-MLM model, the definition is mapped to all related forms, which may correspond to a variety of senses of the target word. This is clearly not ideal, and it may be alleviated by incorporating sense data such as that of SenseBERT [6]. Alternatively, we rather than weighting all related forms equally, we may choose to weight them according to cosine or L2 distance to some contextual embedding of our target word, though we are to this point not sure where we may obtain such embeddings.

Besides, we would also apply our reverse dictionary model to many interesting real-world settings, such as word-crossing, and evaluate their performance. Like the reverse dictionary task, crosswords require translating a multi-word definition into a single-word answer, which is a wonderful application of our reverse dictionary.

6.4 Division of Work

Throughout the project, all our four members contributed actively to the project and we coorporated well in discussing ideas, running experiments, and writing the full report. Specifically, we would like to thank Professor Kemal Oflazer for giving us initial guidance for the project by introducing word embeddings and BERT+MLP model. Our more detailed division of work is as follows (though some of the work are shared):

- **Alex:** Trained the BERT-MLM baseline model and BERT-ML-MLM model; provide the pre-processed the data and code structure; wrote midterm-report and final-report; presentation video.
- **Jinqi:** Trained Character-BERT and Sentence-BERT models; wrote midterm-report and final report; provide insights into the architecture design.
- **Arvin:** Provide insights into the architecture design; presentation slides and video; wrote midterm report and final report.
- **Kai:** Trained WantWords baseline model and neural translation model; wrote midterm report and final report; presentation video.

7 Source Code

All our code and dataset for the reverse dictionary and guidelines of how to run the experiments can be found in the GitHub repository. [7]

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016.

[2] H. E. Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. *arXiv preprint arXiv:2010.10392*, 2020.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] F. Hill, K. Cho, A. Korhonen, and Y. Bengio. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30, 2016.

[5] D. Kartsaklis, M. T. Pilehvar, and N. Collier. Mapping text to knowledge graph entities using multi-sense lstms. *arXiv preprint arXiv:1808.07724*, 2018.

[6] Y. Levine, B. Lenz, O. Dagan, O. Ram, D. Padnos, O. Sharir, S. Shalev-Shwartz, A. Shashua, and Y. Shoham. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online, July 2020. Association for Computational Linguistics.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[8] G. A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[7] `https://github.com/axie66/11785-Reverse-Dictionary-Project`

[9] Y. Morinaga and K. Yamaguchi. Improvement of reverse dictionary by tuning word vectors and category inference. In *International Conference on Information and Software Technologies*, pages 533–545. Springer, 2018.

[10] T. Noraset, C. Liang, L. Birnbaum, and D. Downey. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[11] M. T. Pilehvar. On the importance of distinguishing word meaning representations: A case study on reverse dictionary mapping. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2151–2156, 2019.

[12] F. Qi, L. Zhang, Y. Yang, Z. Liu, and M. Sun. Wantwords: An open-source online reverse dictionary system. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–181, 2020.

[13] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[14] S. Thorat and V. Choudhari. Implementing a reverse dictionary, based on word definitions, using a node-graph architecture. *arXiv preprint arXiv:1606.00025*, 2016.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[16] H. Yan, X. Li, and X. Qiu. Bert for monolingual and cross-lingual reverse dictionary. *arXiv preprint arXiv:2009.14790*, 2020.

[17] L. Zheng, F. Qi, Z. Liu, Y. Wang, Q. Liu, and M. Sun. Multi-channel reverse dictionary model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 312–319, 2020.

8 Appendix

8.1 Qualitative Results

To more directly observe our models, we formulated our own descriptions of words and fed them into our models. They are as follows:

1. a type of tree (multiple targets)
2. medieval social hierarchy where peasants and vassals served lords (feudalism)
3. when someone you trust does something that breaks your trust (betrayal)
4. a very intelligent person (genius)
5. a very smart person (genius)

Note also that we use the unfrozen SentenceBERT and CharacterBERT varieties to generate the predictions below.

OneLook	1. **elm**	2. **pine**	3. **alder**	4. **ash**	5. **maple**
	6. **palm**	7. show	8. hyla	9. **cyathea**	10. tupaia
WantWords	1. **conifer**	2. **maple**	3. **oak**	4. **sycamore**	5. **pine**
	6. **dogwood**	7. **redwood**	8. **sequoia**	9. **cottonwood**	10. **larch**
BERT-MLM	1. **chestnut**	2. **spruce**	3. **pinewood**	4. **teakwood**	5. **linden**
	6. **redwood**	7. **oak**	8. **maple**	9. **logwood**	10. **fir**
BERT (w2v)	1. spp	2. bracts	3. clematis	4. **hollies**	5. shrub
	6. **deciduous**	7. **maples**	8. **conifer**	9. groundcover	10. **dogwood**
BERT-ML-MLM	1. **linden**	2. tree	3. stocked	4. **lime**	5. **teakwood**
	6. **chestnut**	7. **maple**	8. **plum**	9. wood	10. shrub
SBERT	1. tree	2. treed	3. nox	4. canes	5. **pine**
	6. **fir**	7. **maple**	8. arboreal	9. **ironwood**	10. **mahogany**
CBERT	1. **maple**	2. **fir**	3. flag	4. hardwood	5. bole
	6. **pipal**	7. **ironwood**	8. **catalpa**	9. **hickory**	10. **chinaberry**

Table 3: Predictions for "a type of tree". Reasonable predictions are bolded (i.e. predictions such that the phrase "A type of tree is a ___ tree" generally makes sense and is factually correct).

OneLook	1. **feudalism**	2. knight	3. feudal	4. **feudal system**	5. bar
	6. peasant	7. serf	8. aid	9. angel	10. hall
WantWords	1. aristocracy	2. gentry	3. feudal	4. freemasonry	5. **feudalism**
	6. nobility	7. clique	8. monarchy	9. camorra	10. plutocracy
BERT-MLM	1. dukedom	2. fiefdom	3. nobility	4. castle	5. polity
	6. chivalry	7. hierarchy	8. vassalage	9. lords	10. gerontocracy
BERT (w2v)	1. aristocracy	2. feudal	3. monarchical	4. **feudalism**	5. kingship
	6. monarchies	7. noblemen	8. despotism	9. plutocracy	10. caliphate
BERT-ML-MLM	1. oligarchy	2. aristocracy	3. nobility	4. government	5. order
	6. **feudalism**	7. celibacy	8. homage	9. dukedom	10. familly
SBERT	1. serf	2. lord	3. thane	4. feud	5. family
	6. lords	7. fiefdom	8. nobility	9. vassal	10. honoring
CBERT	1. caste	2. lords	3. nobility	4. medieval	5. puritanism
	6. grange	7. feudal	8. druid	9. manor	10. regency

Table 4: Predictions for "medieval social hierarchy where peasants and vassals served lords". Reasonable predictions are bolded. This is from the unseen data.

OneLook	1. confidence	2. reliance	3. want	4. calculate	5. question
	6. trusted	7. leave	8. leaving	9. faith	10. lie
WantWords	1. **betrayal**	2. breach	3. transgressor	4. fraud	5. conning
	6. ratting	7. bigamist	8. breaching	9. mistake	10. cheat
BERT-MLM	1. trust	2. trusty	3. assuring	4. **betray**	5. betraying
	6. judas	7. **betrayal**	8. blackmail	9. compromise	10. trustworthy
BERT (w2v)	1. rapport	2. distrust	3. animosity	4. enmity	5. mistrust
	6. friendship	7. trust	8. remorse	9. affection	10. confide
BERT-ML-MLM	1. trust	2. trusty	3. **betrayal**	4. cheat	5. **betray**
	6. trustworthy	7. faith	8. gamble	9. deception	10. compromised
SBERT	1. trust	2. confidence	3. credit	4. distrust	5. fraud
	6. insurance	7. credited	8. cheat	9. pawn	10. pawning
CBERT	1. adultery	2. fractured	3. fracture	4. fracturing	5. subversion
	6. phishing	7. perfidy	8. history	9. addiction	10. corruption

Table 5: Predictions for “when someone you trust does something that breaks your trust”. Reasonable predictions are bolded.

OneLook	1. brain	2. brains	3. hick	4. hayseed	5. rube
	6. yahoo	7. **brainiac**	8. bumpkin	9. **genius**	10. mastermind
WantWords	1. **savant**	2. **genius**	3. savvy	4. guru	5. **brainiac**
	6. techie	7. brilliant	8. **boffin**	9. brainy	10. intellect
BERT-MLM	1. brainstorming	2. brains	3. brainstem	4. brain	5. brainchild
	6. **brainiac**	7. brainpower	8. **geinus**	9. braincase	10. fool
BERT (w2v)	1. amygdala	2. neurons	3. cortex	4. synapses	5. brain
	6. hippocampus	7. mitochondria	8. neuron	9. macula	10. myelin
BERT-ML-MLM	1. **genius**	2. brains	3. brain	4. intellect	5. einstein
	6. brainstem	7. intelligence	8. brilliant	9. intelligent	10. brainstorming
SBERT	1. brains	2. brain	3. **genius**	4. intellectual	5. intelligent
	6. mandarin	7. wit	8. master	9. monitor	10. mastering
CBERT	1. brains	2. brain	3. **brainiac**	4. lamb	5. savy
	6. brainy	7. intelligence	8. **intellectual**	9. smarting	10. philosopher

Table 6: Predictions for “a very intelligent person”. Reasonable predictions are bolded. The predictions for "intelligent person" and "smart person" actually differ.

OneLook	1. brain	2. clipper	3. spruce	4. stupid	5. dressy
	6. birdbrain	7. **genius**	8. handsome	9. sensible	10. bright
WantWords	1. savvy	2. smarts	3. hotshot	4. techie	5. **savant**
	6. **brainiac**	7. **boffin**	8. savy	9. blaster	10. safecracker
BERT-MLM	1. smart	2. flasher	3. brains	4. smarting	5. flash
	6. smarts	7. brainstorming	8. brain	9. brainstem	10. brainchild
BERT (w2v)	1. suplex	2. dork	3. dropkick	4. tw*t	5. dude
	6. **nerd**	7. skank	8. moron	9. wuss	10. noob
BERT-ML-MLM	1. smart	2. smarty	3. smartness	4. smarting	5. smarts
	6. shoot	7. hothead	8. hotdog	9. flash	10. hotshot
SBERT	1. smarting	2. smarts	3. smart	4. brains	5. brain
	6. **genius**	7. ace	8. intelligent	9. wit	10. dof
CBERT	1. smarting	2. smart	3. smartly	4. smarts	5. savy
	6. sharpie	7. brainy	8. smarty	9. news	10. smartness

Table 7: Predictions for “a very smart person”. Reasonable predictions are bolded. The predictions for "intelligent person" and "smart person" actually differ.