Reverse Dictionary using Neural Models

Arda Unal
unala@uci.edu

Avinash Kumar
avinask1@uci.edu

Hina Arora
hinaa@uci.edu

Abstract

Sometimes it is hard to remember the exact word for something. It can be frustrating to not be able to convey your thoughts because of tip of the tongue phenomenon. We attempt to build a 'reverse dictionary' which maps a given definition or description to the name of the concept. We use neural networks (BOW and RNN) to build two models which map the input definition word embeddings to an output word embedding. Then, we use cosine distance to check for the nearest target words to the output embedding.

1 Introduction

Reverse dictionary is not a new problem in the NLP field. Often, authors and non-native speakers struggle with the inability to find the exact word to convey their thoughts. Instead, they have to settle for a phrasal description or a similar word. This obviously affects the brevity of the work, not to mention the additional doubts created in the mind of the readers due to the lack of appropriate word. This is also called the 'tip of the tongue' phenomenon.

There have been two different approaches towards solving the reverse dictionary problem - 1. Database oriented approach, and 2. Machine Learning Oriented approach. In the database oriented approach, a huge database of word-definition pairs are used to create a graph of words (Thorat and Choudhari, 2016). Then, given an input phrase, a graph based similarity measure is used to get a ranked list of target words. (Gaya, 2015) in their survey paper mention the overall architecture of such models.

In the ML oriented approach too, the concept (meaning) of the input phrase is extracted and vocabulary words are searched for the similar meanings. However, the ML based approaches use neural networks to encode the phrase meaning in word embeddings and then provide candidates with similar word embeddings. The ML based approach is better because of the inherent benefit of using word embeddings (low-dimensional dense vectors) over whole words to define the meaning of words.

The problem of reverse dictionary requires us to map between arbitrary length phrases to fixed length word vectors. We experiment with two broad classes of neural language models- Recurrent Neural Networks (RNNs) which naturally encode the input order of words, and Bag of Words (BOW). The application of these models to this task is described in (Hill et al., 2015). The paper mentions use of pre-trained word embedding obtained using CBOW model. However, we modify the method used by the authors to incorporate the following changes -

- (Hill et al., 2015) used a constant vocabulary set throughout the experiments i.e. training and evaluation. However, in the evaluation phase, we expand our vocabulary to include a much larger vocabulary and their embeddings. This allows our training process to be fast as it has just 100,000 words in its vocabulary. However, in the evaluation phase, we expand the vocabulary to include 418,000 unique words, which become candidates for predictions.

- We also experimented with GRU encoder and GRU decoder with Attention inspired by the last homework. Since we learned about the Neural Machine Translation doing the last homework, we wanted to apply what we learned to this problem by treating the reverse dictionary problem as sequence to sequence

mapping. Even though we experimented with different number of layers for both encoder and decoder as well as with different number of hidden units, results of this approach were not promising. Section 3 explains why this is the case.

- (Hill et al., 2015) used approximately 100,000 words in its vocabulary. In its evaluation sets, when a word wasn't found in vocabulary, the label UNK was used. However, when a word isn't found in the vocabulary, we first stem it and check again. Only when this check also fails, we use the UNK label. This move is aimed at preventing important words from being missed just because they appear in definitions in a different form.

We evaluated the two models (BOW and RNN) on 200 concept descriptions built by people which the authors used in the paper too. These are not dictionary definitions but descriptions for a word that people have come up with. Apart from that, we evaluated our model on 100 seen descriptions. These are descriptions which were a part of the training set. We input these descriptions to our models and picked top 100 candidates. We calculate median rank and rank variance of the correct target word among these 100 candidates.

2 Related Work

We went through several papers which proposed different algorithms for solving the problem. (Gaya, 2015) in their survey paper discuss the overall architecture of a database oriented reverse dictionary system. The input phrases are first preprocessed to undergo stemming and then POS (Parts of speech) tagging. The words are then converted to concepts using thesaurus. The ways to extract concept include distance based measures on the thesaurus. Once the concept is mined, each word in the dictionary is looked up to find definitions which are close to the mined concept. The candidate words are then ranked using probabilistic methods.

(Méndez et al., 2013) propose to solve the problem using an approach similar to word embedding. They choose the top 25 concepts in WordNet and encode the nouns in the input definition into a vector of 25 dimensions based on similarity with each concept. Then, the vectors are averaged and the result is treated as the concept vector of the result. This concept vector is then used to find candidate target words.

(Thorat and Choudhari, 2016) used a distance based similarity measure to evaluate the similarity between the input phrase words and output words. They generated graphs for each input word and similarity to input phrase is measured for every word in the lexicon.

Although all of the above work try to solve the reverse dictionary problem, they use similarity measure based on actual words and do not take into account word embeddings. Recently, word embeddings have shown to inherently capture the meanings of the words in the vector space and the cosine distance between similar words in embedding space are shown to be small. So, we follow the word embedding approach with neural network to solve the problem. We follow the approach described by (Hill et al., 2015) in their paper where they use RNN and BOW.

3 Approach

In this section, we discuss the approach followed by us. First, we describe the dataset and model description as used by (Hill et al., 2015) in Section 3.1, Section 3.2 and Section 3.3. Then, we describe our modifications in the subsequent sections Section 3.4, 3.5 and 3.6.

3.1 Dataset

We obtained the data from (Hill et al., 2015) where the word embeddings for the target words were pre-trained and learned on approximately 8 billion words of running text, using a continuous bag-of-words (CBOW) model with the Word2Vec software. In the above dataset we got, there were a total of 54,722 word embeddings for distinct words with the length of the embeddings being 300. In addition, we got about 900,000 word-definition pairs which the authors had collected from five electronic resources: Wordnet, The American Heritage Dictionary, The Collaborative International Dictionary of English, Wiktionary and Websters. We trained using only about 350,000 of these pairs to let the experiments complete quickly. With 350,000 pairs, 500 epochs of training took about one day for a model. If a word had multiple definitions the word and definition were stored multiple times. We picked top 100,000 unique words from the training set based on frequency to be our vocabulary.

3.2 RNN

As we know, RNNs can operate on variable-length sequences. In our case that would be the dictionary definitions of the training words. The input words of embedding dimension 500 are mapped to an embedding of size 300 of the target word using the RNN. The vocabulary consists of all seen words in the definitions, and for each of these words in the vocabulary a vector v_1 of input embeddings is initialized. The RNN manipulates this embedding as:

$$A_1 = \phi(W'v_1 + b')$$

where W' is weight matrix and b' is bias and ϕ is a nonlinear transformation and A_1 is the first activation state. Then at each epoch the internal activation state is updated:

$$A_t = \phi(U'A_{t-1} + W'v_t + b')$$

RNNs have the limitation that the gradient on the training set can vanish or explode with more iterations. Therefore we use LSTMs with the RNNs where at each time step, six different internal layers are calculated, that summarize the information of the sentence and then a non-linear projection is used to project on to the target embedding space of 300 dimensions.

3.3 Bag-of-words NLMs

Like the RNN case, the simple BOW architecture learns the embedding for all words in the vocabulary with a single weight matrix (W'). The mapping is done using the sum of the product between the weight matrix and the embeddings for each input word. This is a special case of the RNN where U' and ϕ are both identity.

$$A_t = A_{t-1} + W'v_t$$

3.4 Expanding vocabulary

(Hill et al., 2015) used a constant vocabulary set of about 100,000 words throughout the experiments i.e. training and evaluation. However, in the data that authors gave us, there were only about 54,722 words with target embeddings. So, the target set of our system was restrcited to these words only. In order to expand, our target set, we decided to use another set of word embeddings likr Glove or Word2Vec. However, even expanding our embedding set would only increase our target set only to a maximum of 100,000 target words which is the vocabulary size. Here, we made an important change. Both RNN and BOW model output word embeddings. (Hill et al., 2015) compared these word embeddings using cosine distance to all words in vocabulary (for which embeddings are available) and found candidates. However, once we have the target word's embeddings, we need not be restrcited to the vocabulary size. Embeddings have the property that similar words are close together (Glove Embeddings, which have about 400,000 unique words' embeddings, have showed this to be true which is why we use it). Therefore, once we get the target embeddings, we decided to compare it to the entire embedding set. This change increased our system's performance as the median rank predicted by BOW model for concept descriptions fell to 10 from 37. For some reason, RNN model's median rank became worse after this change (jumped to 20 from 9.5). We saw that RNN model had started to predict noisy words like email addresses after this change. However, the number of cases out of 200 concept descriptions for which correct words were identified increased for both RNN and BOW model.

3.5 Encoder-Decoder Model with Attention Decoder

Inspired by the last homework, we experimented with GRU Encoder-Decoder model with Attention to map description sequence to the definition sequence. Even with different number of hidden units and layers, our implemented models map almost all descriptions to the definition word "clear". When we looked our expanded training vocabulary, we have seen that the definition "clear" had 190 different descriptions where as almost half of the labels in the training data have less than 5 definitions. As show below in Figure 1, we have a severe class imbalance and this caused our models to map most descriptions to the major classes reducing the training loss the most. There are some techniques to cope with class imbalance such as undersampling, oversampling and introducing class weights to the batches during training; however considering the long training time, we have decided to improve on our existing RNN and BoW models.

3.6 Using Stemmed words

When a word isn't found in the vocabulary, we first stem it and check again. Only when this check also fails, we use the UNK label. (Hill et al., 2015)

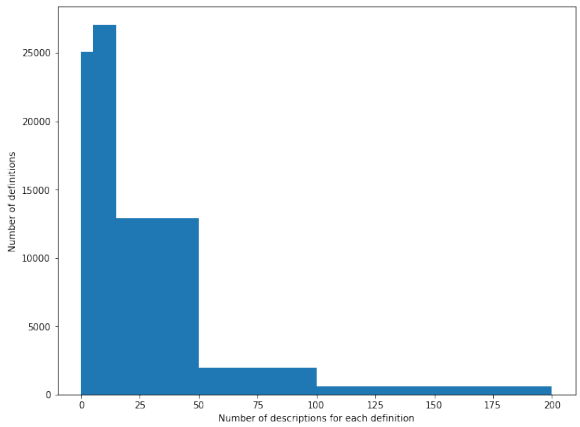

Figure 1: Histogram of labels and how many descriptions they have

	Predictions for *balloons*
Before Stemming	sacking, inflatable, bursting, pneumatic, deflated, rippling, wafted, bowels, banging, permeated
After Stemming	sacking, inflatable, bursting, pneumatic, deflated, banging, overflowing, burst, wafted, screeching

Table 1: Change in predictions by BOW model due to stemming of the word 'childrens'

didn't check for stemmed words. This move is aimed at preventing important words from being missed just because they appear in definitions in a different form. For example, we found that in the concept description for 'balloons', the word 'childrens' is used which is grammatically incorrect. After stemming, the word children is found. The output of BOW model before and after stemming change is shown in Table 1. Although the word balloon doesn't appear, intuitively we can determine that the word 'children' is an important ingredient in describing the word 'balloon'. We couldn't properly test this modifications of ours, because out of the 200 test definitions that we used only 1 definition had one out of vocabulary words. However, we still feel that in real world, this modification will improve the performance.

4 Experiments

(Hill et al., 2015) outlines a comprehensive evaluation plan and our final evaluation process will closely resemble it. We compare our BOW and RNN implementations to the following:

- **Baseline W2V Add** Ranking and returning the nearest word embeddings to the composed vector which is a pointwise summation of the word embeddings for each word in the input query.

- **reversedictionary.org**: Similar to many previous works which had comparisons with commercial online service *OneLook.com*, we compare our implementations to *reversedictionary.org* as we find its performance better over *OneLook.com*. We wrote two Python Selenium scrapers to get results for concept descriptions querying *OneLook.com* and *reversedictionary.org*. Around 40 queried concept descriptions out of 200 returned nothing from *OneLook.com* whereas all queries returned successfully from *reversedictionary.org*.

We evaluate the above and RNN and BOW models on the following evaluation sets:

- **Seen evaluation**: 100 randomly selected words-definition pairs from training set. This evaluation is aimed to measure the recall performance of the models.

- **Concept Descriptions**: In order to have a fair comparison with *reversedictionary.org*, which has both seen and unseen pairs, we use the concept-description dataset (200 word-definition pairs) provided by the authors. These are not dictionary definitions but descriptions for a word that people have come up with. These descriptions are not seen by any of the models.

The results of both models (RNN and BOW) and baselines (W2V add and reverse-dictionary.org) on the two evaluation sets are shown in Table 2. For the two Neural models, we show the results obtained by model suggested by (Hill et al., 2015) and models obtained after our modifications. Note that the rank metrics in Table 2 considers only the cases where target words were found in the predictions. Table 3 lists the number of instances when target words were actually found in the predictions. Table 4 provides a qualitative evaluation of the original and modified BOW models.

Test Set		Seen dictionary definitions (100)		Concept descriptions (200)	
		Median Rank	Rank Variance	Median Rank	Rank Variance
Baseline models	W2V add	NA	NA	43	896
	reversedictionary.org	NA	NA	4	738
NLMs	BOW (Felix et. al)	1	294	37	1214
	BOW (our model)	1	224	10	778
	RNN (Felix et. al)	1	89	9.5	747.85
	RNN (our model)	1	113	20	586

Table 2: Metrics for quantitative comparisons

Test Set		Seen dictionary definitions (100)	Concept descriptions (200)
Baseline models	W2V add	NA	5
	reversedictionary.org	NA	160
NLMs	BOW (Felix et. al)	67	32
	BOW (our model)	83	89
	RNN (Felix et. al)	79	16
	RNN (our model)	94	34

Table 3: Test definitions for which the target word appears in the first 100 candidates.

Table 4: Concept description evaluation of BOW model. We present the top 10 predictions before and after our change of expanding the vocabulary at evaluation stage. The expansion of vocabulary is a major differentiator of our system from the system of (Hill et al., 2015). The highlighted words are those which are close to the actual label.

Concept Description	**Concept Label**	**Predictions before expansion**	**Predictions after expansion**
a room in a house or building where people study or work	office	undercroft, tearoom, refurbished, sunroom, linen, refectory, conservatory, bathroom, townhouse, sacristy	basement, spacious, courtyard, upstairs, hallway, renovated, gymnasium, cafeteria, dormitory, rooms
something that does not cost a lot of money	cheap	overspend expensive, overpriced, scrimp, subsidy, upkeep, profiteering, overpayment, obscene, prepaid	exorbitant, costing, pricey, expensive, subsidize, overpriced, costly, gimmicky, **cheap**, dilettantism
the bottom part of the leg on which people walk	foot	sneakers, romancing, gently, slithery, stubbed, gents, bogie, benni, materialize, muffin	tibia, fibula, femur, forearm, kneecap, gimpy, thigh, instep, ligaments, cheekbone
pieces of information that are true	facts	revelatory, carless, stirred, reckoning, repost, sacrilegious, hooked, superficiality, lovelace, bookshelf	veracity, misinformation, validation, tendentious, disingenuous, factual, cataloging, subterfuge, reportage, validating
the flesh of animals that humans like to cook and eat	meat	overcooked, oxtail, poussin, garlicky, cacciatore, giblet, rarebit, broiled, roulade, mornay	cooked, eaten, eat, **meat**, roast, roasted, chicken, eating, potatoes, vegetables
reduce to make the amount of something lower	reduce	aggravate, deteriorate, impair, stagnate, stabilize, underpin, readjust, constrain, stabilise, overspend	lessen, diminish, lessening, mitigate, minimize, magnify, counteract, optimise, readjust, minimise
an article of clothing that men often wear on their legs	trousers	slouchy, sequined, starched, bandeau, dressy, merkin, bathrobe, porkpie, frilly, jumpsuit	leggings, necktie, miniskirt, tunic, loincloth, overcoat, **trousers**, pinstriped, sparkly, baggy

5 Conclusions and Future Work

We have tried to solve the reverse dictionary problem using RNN and BOW neural network models. We started with the models proposed by (Hill et al., 2015). However, after it became clear that the embeddings we had was not enough, we decided to use Glove embeddings which had embeddings for 400,000 unique words. We made three modifications to the approach of (Hill et al., 2015). The modification, where we expanded our vocabulary, worked very well and improved the performance of our system.

Currently, we are not using all the training data we have due to time concerns. In future work, we can try to incorporate all data. Also, the RNN model's performance deteriorated after the modification of expansion of vocabulary. The model has started to output email addresses as predictions. It would be a good exercise to see why the performance of RNN degrades.

References

D Gaya. 2015. A study of building an reverse dictionary. *International Journal of Scientific and Research Publications* 5(7).

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *CoRR* abs/1504.00548. http://arxiv.org/abs/1504.00548.

Oscar Méndez, Hiram Calvo, and Marco A. Moreno-Armendáriz. 2013. A reverse dictionary based on semantic analysis using wordnet. In Félix Castro, Alexander Gelbukh, and Miguel González, editors, *Advances in Artificial Intelligence and Its Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 275–285.

Sushrut Thorat and Varad Choudhari. 2016. Implementing a reverse dictionary, based on word definitions, using a node-graph architecture. *CoRR* abs/1606.00025. http://arxiv.org/abs/1606.00025.