Reverse Dictionary

A Project Report Submitted by

Aarti Chandrakant Pol

in partial fulfillment of the requirements for the award of the degree of

M.Tech Data and Computational Sciences

Indian Institute of Technology Jodhpur

Mathematics

December, 2021

Declaration

I hereby declare that the work presented in this Project Report titled Reverse Dictionary submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of M.Tech Data and Computational Sciences, is a bonafide record of the research work carried out under the supervision of Dr. Gaurav Bhatnagar. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Aarti Chandrakant Pol

M20MA002

Certificate

This is to certify that the Project Report titled Reverse Dictionary, submitted by Aarti Chandrakant Pol(M20MA002) to the Indian Institute of Technology Jodhpur for the award of the degree of M.Tech Data and Computational Sciences, is a bonafide record of the research work done by her under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr. Gaurav Bhatnagar

Acknowledgements

To start with, I would like to express my sincere gratitude to my supervisor Dr. Gaurav Bhatnagar for supporting me to work in the field of Machine Learning (Natural Language Processing). I would also like to thank him for the guidance and being understanding. I have learnt so many things during this project and am hoping to learn much more ahead. I would like to thank my family for all the love and support. Lastly, I would like to thank the internet due to which I am able to learn so many things and get help whenever I face a problem.

Abstract

Reverse dictionary problem deals with mapping an input query phrase to word(s) which is/are semantically similar to it. The problem is useful both from human and machine learning system point of view. A model that's good at solving the reverse dictionary problem can be helpful while learning new languages, and it can also be helpful in other downstream tasks like machine translation and entity linking. However, in order to solve the problem, the model would need to first learn vector representations of phrases and target words and then learn a function that maps these representations to each other. In this paper, we discuss about learning these vector representations and the models which can be used to solve the problem. We also discuss that current solutions are generally limited to a monolingual setting and how we can move to multilingual and cross-lingual setting for this problem.

Contents

List of Figures

Reverse Dictionary

1 Introduction and background

We all are familiar with a regular or forward dictionary which maps the query words to their definitions. On the other hand, we can also have a reverse dictionary which takes phrase/sentence as input and outputs the target word(s) which is similar in meaning to it. The output might also contain with other semantically similar words to target. Figure 1.1 shows example of a forward and reverse dictionary on a given set of words. Like, forward dictionary, a reverse dictionary has many day-to-day use cases:

- It can aid with vocabulary while learning a new language.
- It can help with "tip-of-the-tongue" problem that we all face from time to time where we know the sentence but couldn't remember the word for it.
- It can also be helpful in case of some neurological disorders where people are able to identify and describe an object but struggle to name it.

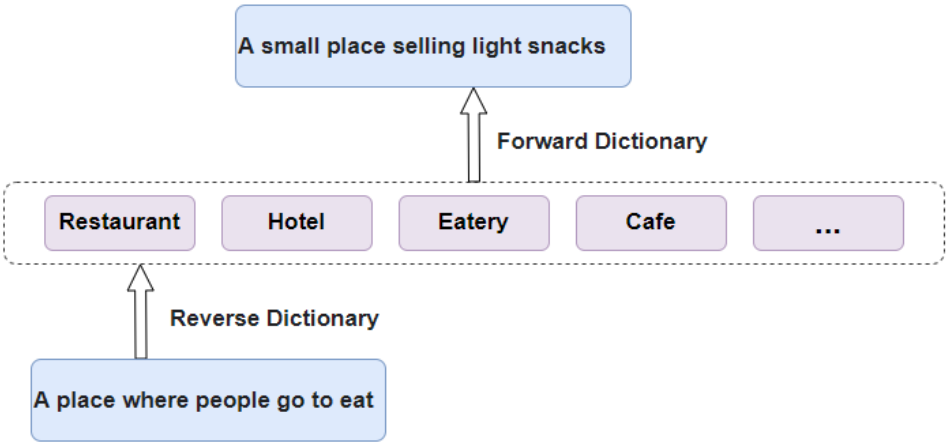

Figure 1.1: Example of forward and a reverse dictionary

Apart from above usefulness, reverse dictionary models can be helpful in downstream machine learning tasks like evaluating the quality of phrase/sentence representations, machine

translation etc. as well. It can also be useful for tasks like question answering and entity linking/mapping.

Given the importance of the problem, there are a few existing in-production systems for the reverse dictionary. Two prominent ones are OneLook.com and ReverseDictionary.org which use a variety of methods in the backend to solve the problem. In Figure 1.2 and 1.3, we can see the output from these systems respectively for the entered query phrase.

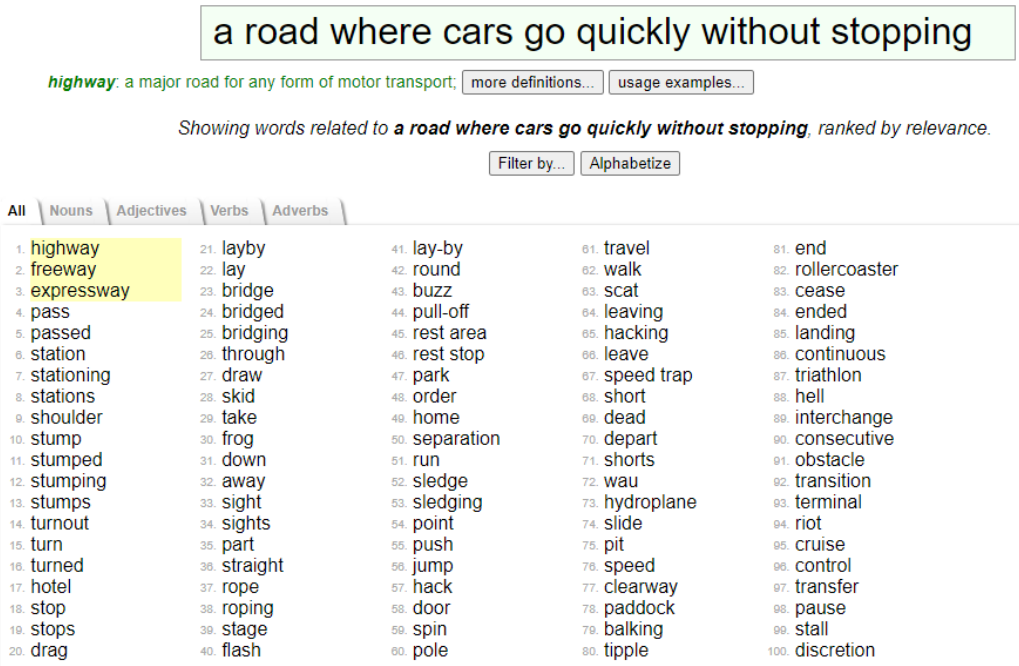

Figure 1.2: Output of query obtained from OneLook.com

1.1 Limitations of these products

Though the online products perform decently on the task of reverse dictionary, they are not open source so there is lack of clarity about ythe algorithms and models they are using to solve the problem. These systems are not multilingual, so if the input sentence is in language other than English (say Hindi, Marathi etc.), these products won't be able to give the corresponding target word for them. Similarly, they won't work in cross-lingual setting as well, for e.g. when the input sentence is in language $L1$ and we would like to know the corresponding target word in language $L2$.

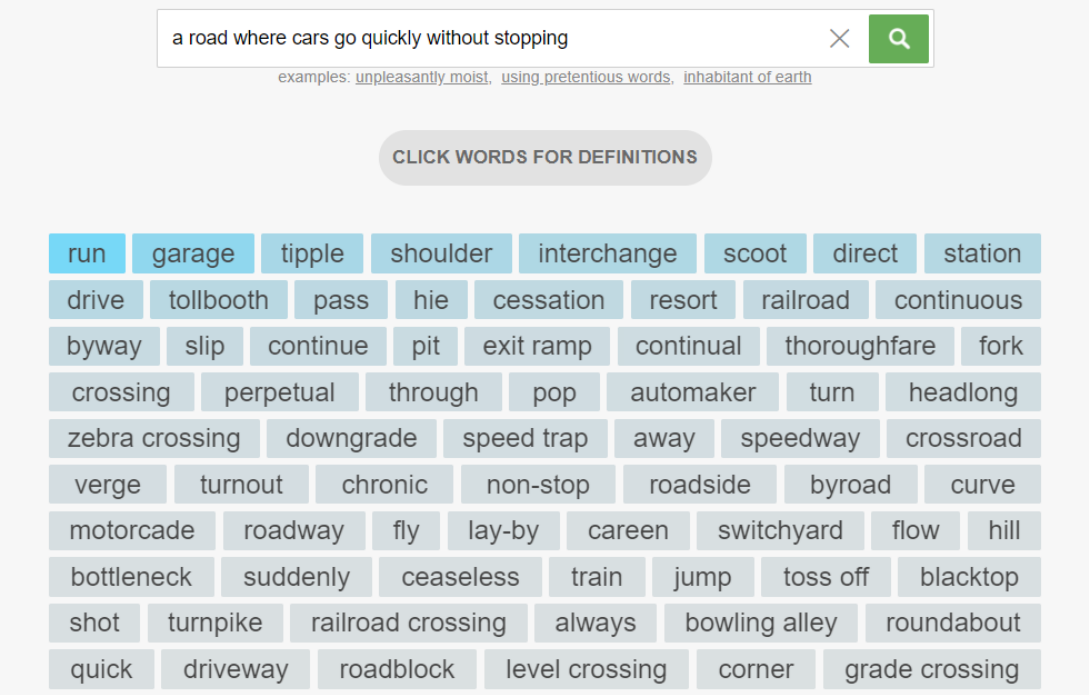

Figure 1.3: Output of query obtained from ReverseDictionary.org

2 Definitions

Natural language processing (NLP) is an interdisciplinary field of computer science, linguistics and artificial intelligence which deals with the understanding of human natural language and their interaction with computer systems. Some of the major problems in this domain include machine translation, text classification, text summarization, entity recognition, questions answering, dialogue engine etc.. In the past, statistical analysis was used to tackle some of these problems while others were not approachable. However, with the rise of various deep learning methods during the last decade, there has been tremendous progress in the NLP field. In this section, we will go over some of the terms which will be helpful in understanding the problem statement and modelling approaches.

2.1 Text Pre-processing

In any NLP task, we start with pre-processing the given text data. There are various steps involved in text pre-processing as shown in 2.1.

- **Tokenization:** It is a process of converting given input sentence into independent words

(called as tokens). It's generally the first step in an NLP pipeline. First, we chunk/split the given input data (corpus) into sentences and then these sentences are split into tokens. Depending on the input text, various preprocessing methods need to be applied to get the desired output.

- **Stemming:** It is a process by which we convert a token to it's stem form by removing the prefix/suffix/infixes/circumfixes. It works at token level.
- **Lemmatization:** This process aims to return the base or dictionary form of a token which is called a $lemma$. It depends on the correct identification of intended meaning of the token. It works at token as well as context level.
- **Stop Words:** These are the common words which are filtered out during processing of text. These are generally most commonly occurring words in data (like 'a', 'an', 'the' etc.) and generally don't contribute much to the meaning of text.
- **Parts-of-speech (POS) Tagging:** As the name suggest, POS tagging implies assignment of part-of-speech tags like 'noun', 'verb', 'adjective' etc. to the tokens in the sentences.

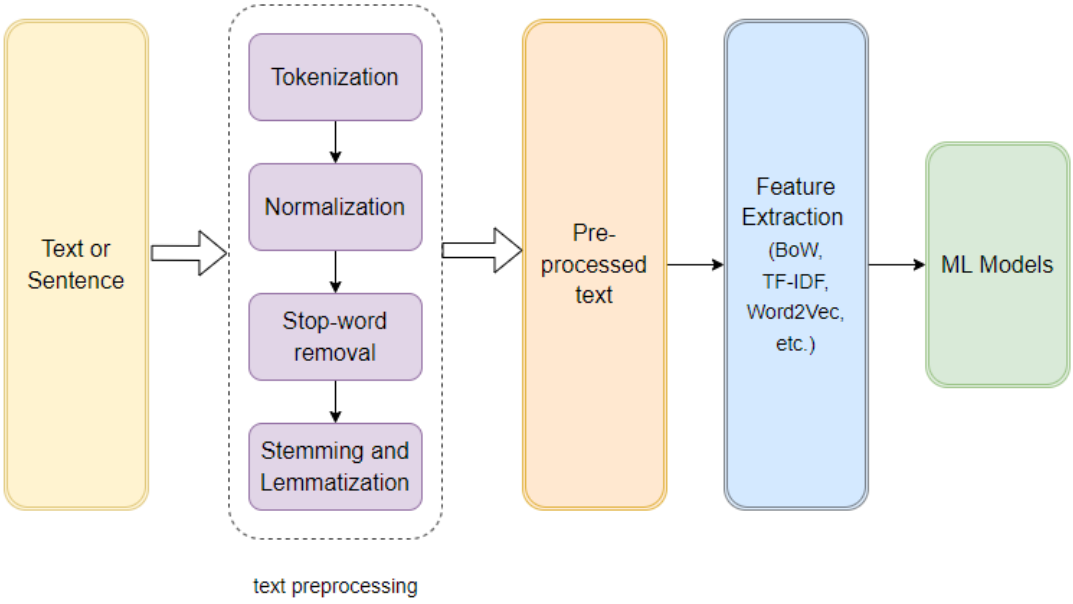

Figure 2.1: NLP Pipeline

2.2 Text featurization

Computer algorithms don't work with words or letters rather they work with their numerical representations. In the following section, we discuss some of the common techniques that convert text to their corresponding numerical representations. We can perform the mathematical operations on these representations.

- **One-hot encoding:** It is the simplest way to represent the words numerically where every word is represented by a sequence of $0s$ and $1s$. The dimension of each word representation will be equal to the number of unique words in the corpus.

- **Bag of Words (BoW):** It is an extension of one-hot encoding to represent text. In BoW, given text is represented by a $n * V$ matrix where n is the number of documents and V is the number of unique words in the whole corpus.

- **TF-IDF Model:** This model tries to improve upon the BoW model by incorporating information about more and less important words in the vector. It's made up of two terms - tf and idf. Term frequency (tf) measures the frequency of words in a document. On the other hand, inverse document frequency (idf) measures the inverse frequency of the number of documents containing the word.

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

$$idf_t = log\frac{Number\ of\ documents}{Number\ of\ documents\ with\ term\ 't'}$$

$$(tf - idf)_{t,d} = tf_{t,d} * idf_t$$

While both the BoW and TF-IDF model allows us to represent the given text into a vector form, they don't capture the meaning for words and sentences themselves i.e. by using these representations we can't say if word "computer" will be closer to word "science"

rather than word “mango”. This is where word embedding based methods excel. We will discuss about some of these methods in following sections.

mango	1	0	0	0	0
is	0	1	0	0	0
sweet	0	0	1	0	0
and	0	0	0	1	0
pulpy	0	0	0	0	1

	and	is	mango	orange	puppy	sour	sweet
mango is sweet and puppy	1	1	1	0	1	0	1
orange is sweet and sour	1	1	0	1	0	1	1

Figure 2.2: The left figure shows the output of one-hot encoding vector and the right figure shows BoW output

3 Word Embedding

Word embedding is a modelling technique where words or phrases are embedded (mapped) to real valued vectors such that their meaning is captured. Embedding in itself implies mapping a high-dimensional vector into a low-dimensional space while placing the semantically similar words closer in the embedding space.

Unlike BoW and TF-IDF, in word embedding these vectors are learnt by optimizing some objective function which ensures that they capture the meaning of the words they are representing. Main idea of these models is that words which have similar neighbors (context) will have similar meanings and hence they have similar vector representation in the embedding space as shown in Figure 3.2. This is also known as “distributional hypothesis or *”You shall know a word by its surrounding words”*.

Once the vectors are learnt, we can apply mathematical operations like dot product on word vectors to measure their similarity. We can also perform addition and subtraction of word embeddings to solve word analogy tasks. Some commonly used word vectors come from these embeddings Word2Vec [2] , GloVe[3], FastText[4, 5]. Word2Vec and GloVe are word based embeddings while FastText is subword based i.e. it can represent a word using its subword when the representation for the whole input word is not present.

The two common methods of learning word embeddings are Continuous Bag of Words (CBOW) and Skip-gram which are shown in Figure 3.1.

- **In CBOW**, we take the context vector C of each word as the input and try to predict the word corresponding to the context. The hidden layer is the weighted sum of input vectors. We apply a softmax function on hidden layer output to get probability distribution over the vocabulary.

- **In Skip-gram**, we use the target word (whose representation we want to generate) to predict the context words. We input the target word vector into the network and get C probability distributions (representing each context word) over vocabulary V.

  In both cases, we take the argmax over the probability distribution to get the predicted word. CBOW is faster to train while skip-gram works well with small amounts of data and represents rare words better[6].

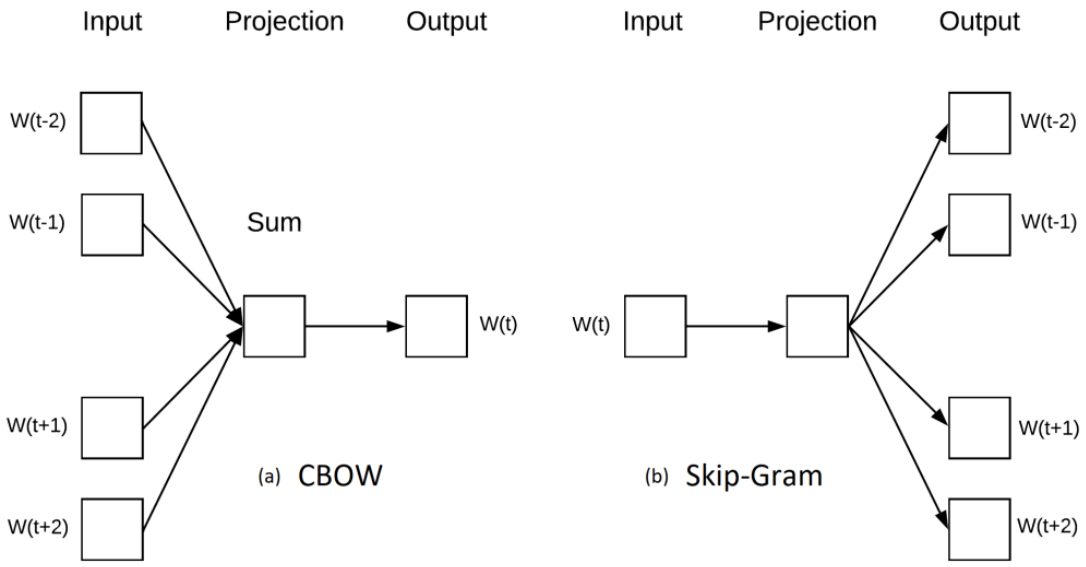

Figure 3.1: CBOW and Skip-Gram models

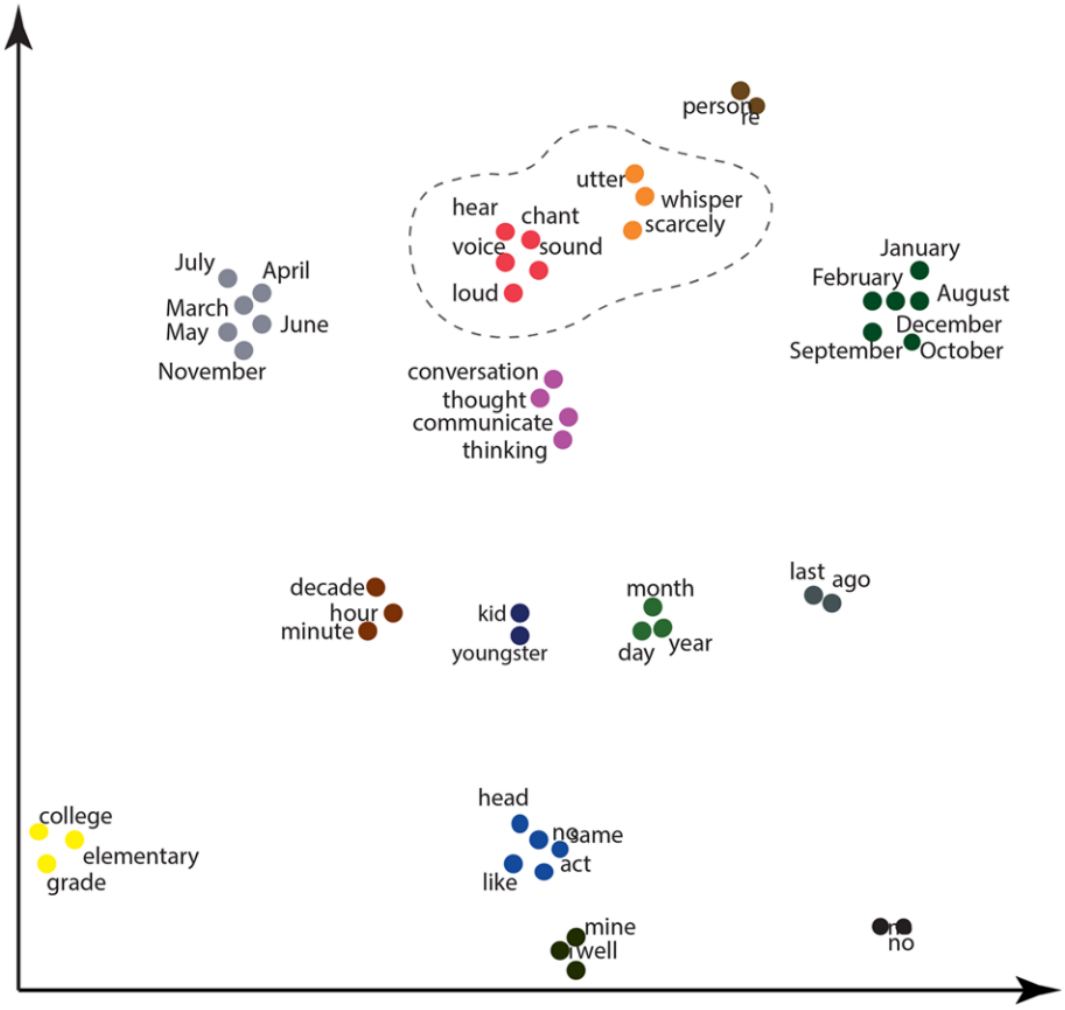

Figure 3.2: Word embeddings represented in a $2d$ space

4 Neural Language Model (NLM) Architecture

Hill et al., TACL2016[1] introduced the problem of reverse dictionary. They suggested Long Short Term Memory (LSTM) based approach to solve the problem. LSTM models are quite good at handling sequential data. An LSTM cell control the flow of information using three types of gates - input, forget and output. These gates are neural networks in themselves and acts as filters. At each timestep, the output of LSTM is dependent on three things - memory cell state (current long-term memory of the network), previous hidden state (output at previous timestep) and input at current timestep as shown in figure.

The authors passed the input sentence S to LSTM model M to get it's context vector (final

hidden state) $M(S)$. Then they train M to reduce the distance between the vector $M(S)$ and the pre-trained embedding of target word emb_w using cosine embedding loss. This objective allowed the model to bring the representation of phrase closer to representation of the target word. During inference time, we return the words whose embeddings are closest to the input phrase vector representation.

One of the drawback of this model was that many words are of low-frequency as per Zipf's Law[7] and thus have poor embeddings. Since the model is trained on the objective is to bring the representation of $M(S)$ and emb_w closer, it will be adversely affected by words which have poor representation.

5 Multi-channel Reverse Dictionary Model

Zhang et al., AAAI 2020[8] proposed a model inspired by how humans tries to infer the word from a given description. For e.g. if we forget what's the meaning of word "expressway", we might use the fact that it's a noun. We might also use the fact it's an entity and an abstract emotive word. Similarly, we can also guess that the target word might contain "way" as morpheme. Thus, the example shows that having an overall knowledge about the characteristic of target word makes it much easier to search for it. So they proposed a model that has multiple predictors to identify different characteristics of target words from input phrase as shown in Figure 5.1.

5.1 Characteristics Predictors

In order to identify different characteristics of target words, they use multiple predictors from input queries. This helps with the target words that have poor embeddings as they can still be predicted due to their characteristics. It also helps to filter words that have close embeddings to the correct target word but have contradictory characteristics to the given description. Each characteristics predictor act like an information channel searching the target word.

- **Internal channels:** These represents the characteristics of words/tokens such as POS tag and morpheme. Morpheme is the smallest meaning unit of a word for e.g. word "unbreakable" has three morphemes - {"un", "break", "able"}.

- **External channels:** These represent the characteristics of target words such as word category and sememe which can be linked using external knowledge bases. Sememes represents meaning of a word at atomic/morpheme level, for e.g. morpheme ”un” means ”not”.

The authors used a BiLSTM model with attention layer as a base model and then applied four characteristic predictors described below to it. The confidence score $sc_{w,word}$ of each word w is calculated using dot product with input query vector(v).

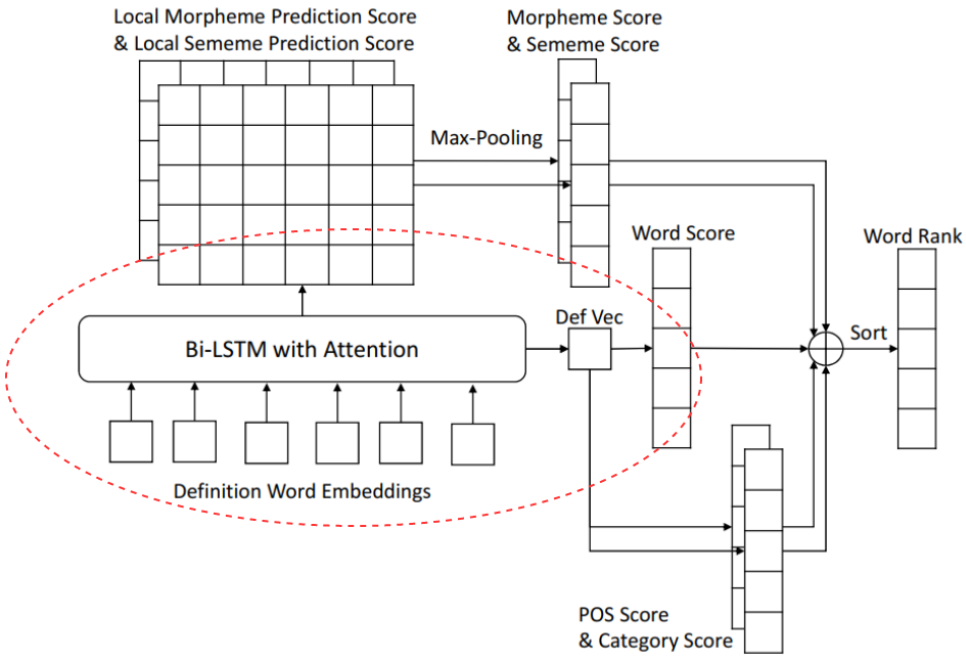

Figure 5.1: Multi-channel Reverse Dictionary model. Dotted red ellipse is similar to the model used in NLM work by Hill et al., TACL2016[1]

$$\mathbf{v}_{word} = \mathbf{W}_{word} * \mathbf{v} + \mathbf{b}_{word} \tag{1}$$

$$sc_{w,word} = \mathbf{v}_{word}.\mathbf{w} \tag{2}$$

5.2 Internal Channels

5.2.1 POS Tag Predictor

A model that is trained to predict the pos tag target word would ensure that it doesn't return target words with a different pos tag that input query. This is achieved by passing the context vector v of the input query to a single-layer perceptron. The confidence score sc_{pos} of word w comes from the POS tag channel and is the sum of the prediction scores of w's POS tags.

$$sc_{pos} = \mathbf{W}_{pos} * \mathbf{v} + \mathbf{b}_{pos} \tag{3}$$

5.2.2 Morpheme Predictor

As mentioned above, morpheme is the smallest meaningful unit of a word. The authors used Morfessor [9] to segment all the words into morphemes. For the morpheme predictor, each hidden state is used and then max-pooling is applied to get final predictions. The task of predicting morphemes of the target word using the input vector representation helps the model to learn target word's compositional information. sc^i_{mor} denotes the semantic correspondence between i^th word in the input query and each morpheme.

$$sc^i_{mor} = \mathbf{W}_{mor} * \mathbf{h}_i + \mathbf{b}_{mor} \tag{4}$$

5.3 External Channels

5.3.1 Word Category Predictor

Even if two words are semantically related and have similar word vectors, they can belong to different categories. For e.g. words "car" and "road" are semantically related but belong to different categories. The authors used WordNet [10] for annotate the input phrase words with their category. Word category predictor is designed similar to POS tag predictor. As the model learns to predict word category, it has the ability to remove semantically related (to input) but not similar words from results.

5.3.2 Sememe Predictor

As discussed, sememe represents meaning of a word at atomic/morpheme level. The authors used used HowNet [11] to get sememe information for input phrase. Sememe of word has local semantic correspondence with it's description. For e.g. sememes for "expressway" is {"route", "fast"} and both are semantically closer to words "road" and "quickly" in the definition of the word i.e. "A road where cars go very quickly w/o stopping". Sememe predictor is designed similar to the morpheme predictor.

Final confidence scores (probability) of target word w is obtained by combining confidence scores from both the direct word prediction task and four indirect characteristic prediction tasks.

$$sc_w = \lambda * sc_{w,word} + \sum_{c \,\epsilon\, pos,mor,cat,sem} \lambda_c * sc_{w,c} \tag{5}$$

Overall, the authors showed that having knowledge of various characteristics of words in the input phrase makes it much easier to search for the target word. Hyperparameters λ_{word} and λ_c control the relative relative weights of corresponding terms and all the parameters are trained using the cross-entropy loss.

6 Problem definition and Objective

The problem of reverse dictionary involves mapping a given input sentence to the word(s) which are similar in meaning to it. Humans are quite good at solving this problem due to understanding of the world around them. For a machine (an NLP model here), to be able to solve this problem, it has to understand the query sentence semantically and then be able to map the sentence to word present in vocabulary. Thus the problem involves learning good vector representations of input phrases as well as word vectors. Afterwards, we need to learn a way to learn a model that can do $M -> M$ mapping of input phrase vector to word vectors.

As discussed in previous sections, most existing methods have tried to solve the reverse dictionary problem when both input and target belong to one language (mainly English). Our goal is to build a model that can can solve the problem in a multilingual setting i.e. when input/target can be multiple languages like Hindi, Marathi and so on. Further we would like to extend the model to a cross-lingual setting where input and target can be in different languages. Figure 6.1 shows both such settings on a set of three languages.

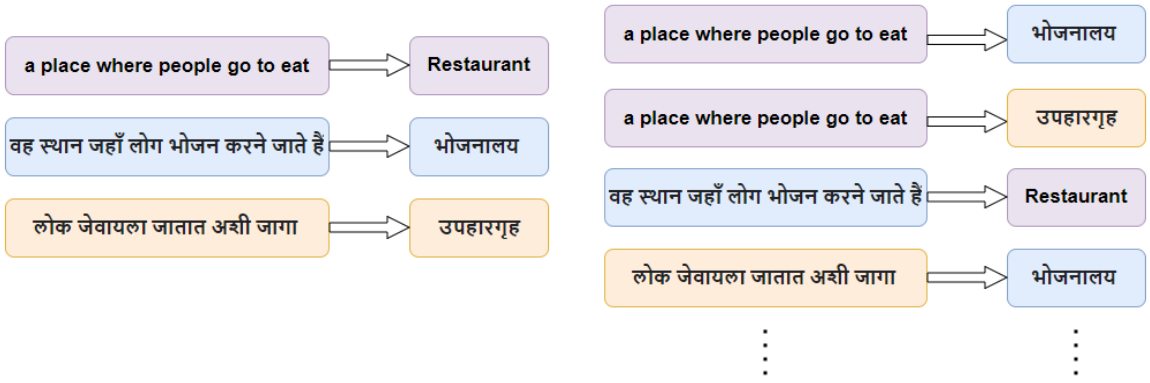

Figure 6.1: Figure on left side shows monolingual reverse dictionaries and Figure on right shows cross-lingual reverse dictionaries for three languages.

7 Summary and Future plan of work

In this paper, we discuss an interesting problem called reverse dictionary from the domain of NLP, solving which requires learning good representation of phrases/words and then learning to map these representations. We see how a model that can solve this problem be useful in many day-to-day task like helping with learning new language or solving crossword puzzles. In section 2, we discuss various pre-processing methods that are applied to input text before passing it to a model. After that, we see what an embedding is and what are some of the common approaches to learn word embeddings. Section 4 and 5 discuss methods to solve the reverse dictionary problem using an LSTM based neural network in a monolingual setting.

As discussed in previous section, our main objective is to solve the reverse dictionary problem in a multilingual and cross-lingual setting as shown in Figure 6.1. To achieve this goal, we would have to collect parallel training data in various languages. Specifically, we would need tuples of the form $\{phrase_{l1}, target\_word_{l1}\}$, $\{phrase_{l2}, target\_word_{l2}\}$ etc. in case of multi-lingual setting and $\{phrase_{l1}, target\_word_{l2}\}$, $\{phrase_{l2}, target\_word_{l3}\}$ etc. in case of cross-lingual setting. Afterwards, we would have experiment with various word and phrase embedding methods in these languages.

From the computation point of view, the problem has another challenge i.e. it would involve learning models for each phrase and target language pair. So, if want to have a model that works on n languages, essentially that would mean learning $n * n$ sub-models for each language pair. This is prohibitive in terms of memory to save these models and also in terms time of taken to train models individually. We would like to experiment with approaches that can help solve the problem with single or lesser number of sub-models such as multi-lingual word embeddings and transformers.

References

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

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[5] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[6] "word2vec code archive - https://code.google.com/archive/p/word2vec/."

[7] "Zipf's law: Modeling the distribution of terms - https://nlp.stanford.edu/irbook/html/htmledition/zipfs-law-modeling-the-distribution-of-terms-1.html."

[8] 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*, vol. 34, no. 01, 2020, pp. 312–319.

[9] T. Ruokolainen, O. Kohonen, S. Virpioja, and M. Kurimo, "Painless semi-supervised morphological segmentation using conditional random fields," 01 2014, pp. 84–89.

[10] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[11] Z. Dong and Q. Dong, “Hownet - a hybrid language and knowledge resource,” in *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, 2003, pp. 820–824.