UPPSALA
UNIVERSITET

Palindromes

Never odd or even

Shifei Chen

Uppsala University
Department of Linguistics and Philology
Master Programme in Language Technology
Master's Thesis in Language Technology, 30 ECTS credits

June 19, 2020

Supervisors:
Ali Basirat, Uppsala University

Abstract

The concept of *palindromes* is introduced, and some method for finding palindromes is developed.

Contents

Preface

This thesis was finished under the supervision from Ali Basirat. I would like to thank him for his continuous help and inspriration.

I would like to thank Mr. Anders Wall and everyone in the Anders Wall Scholarship Foundation for sponsoring my Master study. I would also like to thank everyone in the Master Programme in Language Technology, including all of my classmates and the teachers. I have learned a lot from you during this 2-years journey.

Last but not least, I would like to say thank you to my parents for their unconditional love and support. Also to my girlfriend, who has always been together with me during this unusual time.

1 Introduction

Palindromes are fun. I've tried to find some. In Chapter 2 previous work is reviewed, and Chapter 4 is about my results.

2 Previous work

2.1 Word Embeddings

2.1.1 Representing Words in Vectors

In Natural Language Processing, people need to convert the natural representation of words into form that are more effieicent for computer to process. The idea started with statistical language modelling (Bengio et al., 2003). In 2013, Mikolov, Chen, et al., 2013 introudced Word2Vec, which encapsules words and their latent information into vectors. Besides the benefit that it simplifies representation and storage of words for computers, it also enables the possibilities to calcualte word and their semantic meanings just as vectors.

Take an example vocabulary $V = \{\text{king}, \text{queen}, \text{man}, \text{woman}\}$, if we convert these words into vectors such as

$$\begin{aligned}\vec{k} &= \text{vec}(\text{king})\\ \vec{q} &= \text{vec}(\text{queen})\\ \vec{m} &= \text{vec}(\text{man})\\ \vec{w} &= \text{vec}(\text{woman})\end{aligned}$$

We could have an equation of

$$\vec{q} = \vec{k} - \vec{m} + \vec{w} \tag{2.1}$$

It is meaningful from both the mathmatical prospective and the linguistic prospective. The latter can be illustrated by Figure 2.1 in a vector space that contains these four vectors. In addition, the two cosine similarity values of vectors $\vec{k}$ and $\vec{q}$, and of $\vec{m}$ and $\vec{w}$ should also be close, as the angles between each two vectors are about the same.

To turn words into vectors, one could use simple one-hot encoding. Like in the example above we could make $\vec{k} = [1, 0, 0, 0]$. But these one-hot vectors can merely

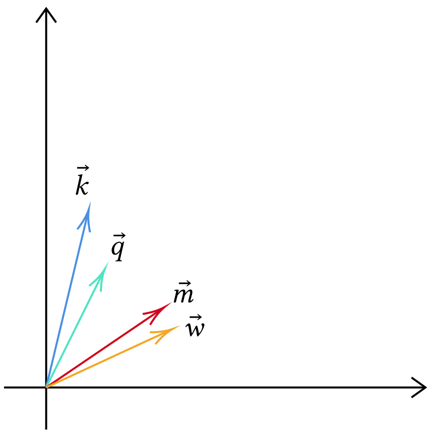

Figure 2.1: Illustraion of a vector space where Equation 2.1 exists.

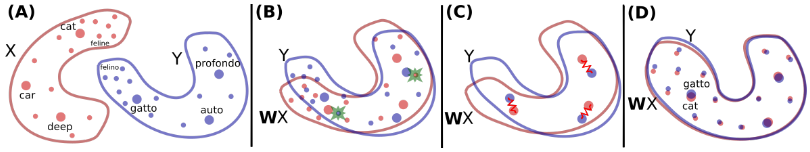

Figure 2.2: Aligning bilingual vector spaces. (Conneau et al., 2017)

capture any latent semantic meanings between different words. Recent vectorized word representations, or word embeddings, were learned through neural networks, such as Word2Vec which learns word embeddings through a Skip-gram model or a Continuous Bag of Words model (Mikolov, Sutskever, et al., 2013).

The Skip-gram model

When given a target word w, the model can produce vector representations that are good at predicting the words surrounding w within the context size of C. The probability of a context word w_k given a target word w is:

$$P(w_k|w) = \frac{\exp({v'_{w_k}}^\top v_w)}{\sum_{i=1}^{|V|} \exp({v'_i}^\top v_w)} \tag{2.2}$$

Here $|V|$ means the size of the whole vocabluary from the corpus, v' and v stand for the vector representation of the input and the output vector representation of a word (Mikolov, Chen, et al., 2013). The input representation v' could be initialized by one-hot representations.

The Continuous Bag of Words model (CBOW)

The other model, CBOW, works just as the other side the coin. It predicts the target word w based on a bunch of context words $w_{-C}, w_{-C+1}..., w_{C-1}, w_C$ within the window size C, as the formula below:

$$P(w|w_{-C}, w_{-C+1}..., w_{C-1}, w_C) = \frac{\exp({v'_w}^\top \bar{v}_{w_k})}{\sum_{i=1}^{|V|} \exp({v'_{w_i}}^\top \bar{v}_{w_k})} \tag{2.3}$$

Here $\bar{v}_{w_k}$ means the sum of the context word w_k's vectorized representation, while v'_w means the input vector representations of word w as in the Skip-gram model.

The difference between these two models is that the CBOW model predicts the target word from multiple given context words, while the Skip-gram model predicts the context words from one given center word. Hence the skip-gram model is better at predicting rare words because all of the words are treated equally in the *word AND context* relationship. But in the CBOW model, common words have advantages over rare words as they will have higher probability in a given context. The Skip-gram model is arguably the most popular method to learn word embeddings as it is both fast and robust (Levy et al., 2015).

2.1.2 Cross-Lingual Word Embeddings

Vectorized word representations tends to cluster words that are semantically similar to each other. It then become very attractive to see whether we could fit two or more langauges into the same vector space. This is so called multilingual word embeddings.

In such case, it is then vital to align words in two different vector spaces. As show in Fig. 2.1.2, which illustrated the alignment method from Conneau et al., 2017. Suppose there is a set of word pairs in their associated vertorized representation $\{x_i, y_i\}_{i\in\{1,...,n\}}$, the two vector spaces were aligned by learing a rotation matrix $W \in \mathbb{R}^{d \times d}$ as in process **(B)**, where we try to optimize the the formula

$$\min_{W \in \mathbb{R}^{d \times d}} \frac{1}{n} \sum_{i=1}^{n} \ell(Wx_i, y_i) \tag{2.4}$$

. Here ℓ is the loss function and it is usually the square loss funciton $\ell_2(x, y) = ||x - y||^2$. W is then further refined in process **(C)**, where frequent words were selected as anchor points and the distance between each corrospondent anchor points were minimized by using an energy function. After this, the refined W is then used to map all words in the dictionary during the inference process. The translation $t(i)$ of a given source word i is obtained in the formula

$$t(i) \in \operatorname*{arg\,min}_{j \in \{1,\ldots,N\}} \ell(Wx_i, y_j) \tag{2.5}$$

. Again, the loss function ℓ is typically the square loss function. However using square loss could make the model suffer from the "hubness problem". Conneau et al., 2017 counter reacted to the "hubness" problem by introducing the cross-domain similarity localscaling (CSLS).

The initial alignment data to for adversarial learning the rotation matrix W could come from a bilingual dictionary (Mikolov, Le, et al., 2013). There are other kinds of alignment by using aligned data from sentence level, or even document level. By using word-level information, we can start with a pivot lanugage (usually English) and map each other monolingual word embeddings by looking up translation dictionaries. This could also be done starting with bilingual vector spaces, where we choose a bilingual word embedding that shares a language (typically English) with other bilingual embeddings, and choose other bilingual word embeddings by aligning their shared language subspace. Sentence-level parallel data are similar data as the corpus in Machine Translation (MT), which contains sentence-aligned texts (Hermann and Blunsom, 2013). Document-level information are more common in the form of topic-aligned or class-aligned, such as Wikipedia data (Vulić and Moens, 2013).

The alignment process of multilingual word embeddings are roughly the same as bilingual word embeddings, using parallel data from either word-level, sentence-level or document-level (Ruder et al., 2019).

2.1.3 fastText

In this work, we have chosen fastText aligned word vectors [1] (Joulin et al., 2018) as our vectorized word representation. They are based on the pre-trained vectors computed on Wikipedia using fastText (Bojanowski et al., 2016).

fastText is an extension to the original Word2Vec methods which uses sub-words to augment low-frequency and unseen words. For example, `low-key` as a whole word its possibility in a given document would be much lower than each of the component, `low` and `key`. fastText learns its vectorized representation from a smaller n-gram sub-word level. It divides the whole word into sub-words units as below if we assume $n = 3$

`<lo`, `low`, `ow-`, `w-k`, `-ke`, `key`, `ey>`

Each of the sub-word has its own vectorized representation learned through a CBOW or Skip-gram model as in Word2Vec. The word vector for the whole word unit `<low-key>` is then the sum of all of its sub-word units' vectors, hence its rareness

[1] https://fasttext.cc/docs/en/aligned-vectors.html

would be compensated by two rather frequent subwords `low` and `key`, even if it might not appear in the training document at all.

In terms of multilingual alignement, fastText improves the common solution to the hubness problem by directly including the Relaxed CSLS (RCSLS) criterion into the model during both the learning and the inference phrase. Before the work of Joulin et al., 2018, inverted softmax (ISF) Smith et al., 2017 or CSLS (Conneau et al., 2017) was only used in the inference time to address the hubness problem while square loss is still the loss function used in the training time. But since both the ISF and the CSLS are not consistent with the square loss function in the training time, they will create a discrepancy between the learning of the translation model and the inference.

2.1.4 Multilingual Neural Machine Translation Systems (Using Aligned Word Embeddings)

One of the potential application of multilingual word embeddings is machine translation. In cases where people need to translate from or into a low-resource langauge, they usually find it difficult to locate enough parallel data that consists of such kind of less common language. If we could build up a vector space with word embeddings from different languages that are aligned, we could leverage the similarity of word embeddings to compensate the lack of parallel data (Zou et al., 2013). We could find words that are never seen in the training data buy looking for their neighbours in the vector space. There are case where successfully trained a machine translation system using very little or none parallel data (Conneau et al., 2017).

2.2 Language Embeddings

2.3 Machine Translation

3 Methodalogy

4 Results

5 Analysis

Bibliography

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (2003). "A Neural Probabilistic Language Model". *J. Mach. Learn. Res.* 3.null (Mar. 2003), pp. 1137–1155. ISSN: 1532-4435.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov (2016). "Enriching Word Vectors with Subword Information" (July 2016). eprint: 1607.04606. URL: https://arxiv.org/pdf/1607.04606.pdf.

Conneau, Alexis, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou (2017). "Word Translation Without Parallel Data" (Oct. 2017). eprint: 1710.04087. URL: https://arxiv.org/pdf/1710.04087.pdf.

Hermann, Karl Moritz and Phil Blunsom (2013). "Multilingual Distributed Representations without Word Alignment" (Dec. 2013). eprint: 1312.6173. URL: https://arxiv.org/pdf/1312.6173.pdf.

Joulin, Armand, Piotr Bojanowski, Tomas Mikolov, Herve Jegou, and Edouard Grave (2018). "Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion" (Apr. 2018). eprint: 1804.07745. URL: https://arxiv.org/pdf/1804.07745.pdf.

Levy, Omer, Yoav Goldberg, and Ido Dagan (2015). "Improving Distributional Similarity with Lessons Learned from Word Embeddings". *Transactions of the Association for Computational Linguistics* 3, pp. 211–225. DOI: 10.1162/tacl_a_00134. URL: https://www.aclweb.org/anthology/Q15-1016.pdf.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient Estimation of Word Representations in Vector Space" (Jan. 2013). eprint: 1301.3781. URL: https://arxiv.org/pdf/1301.3781.pdf.

Mikolov, Tomas, Quoc V. Le, and Ilya Sutskever (2013). "Exploiting Similarities among Languages for Machine Translation" (Sept. 2013). eprint: 1309.4168. URL: https://arxiv.org/pdf/1309.4168.pdf.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality" (Oct. 2013). eprint: 1310.4546. URL: https://arxiv.org/pdf/1310.4546.pdf.

Ruder, Sebastian, Ivan Vulić, and Anders Søgaard (2019). "A Survey Of Cross-lingual Word Embedding Models". *JAIR* 65, pp. 569–631. DOI: 10.1613/jair.1.11640. eprint: 1706.04902. URL: https://arxiv.org/pdf/1706.04902.pdf.

Smith, Samuel L., David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla (2017). "Offline bilingual word vectors, orthogonal transformations and the inverted softmax" (Feb. 2017). eprint: 1702.03859. URL: https://arxiv.org/pdf/1702.03859.pdf.

Vulić, Ivan and Marie-Francine Moens (2013). "A Study on Bootstrapping Bilingual Vector Spaces from Non-Parallel Data (and Nothing Else)". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1613–1624. URL: https://www.aclweb.org/anthology/D13-1168.pdf.

Zou, Will Y., Richard Socher, Daniel Cer, and Christopher D. Manning (2013). "Bilingual Word Embeddings for Phrase-Based Machine Translation". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1393–1398. URL: https://www.aclweb.org/anthology/D13-1141.pdf.