A WSD System and a POS Tagger for a Rare Language

Mingxu TAO
School of Electronics Engineering and Computer Science
Peking University
kuroko@pku.edu.cn

Abstract

In the paper, we introduce a method to transfer words to vectors at first. Based on this word-to-vector method, we detail a translation model to judge whehter a sentence in English and a sentence in this rare language have the same meaning or not. Then, we can use the result to build a WSD and a POS system.

1 A Word to Vector Model

In this section, we will introduce a model to encode words in vectors. Different from some `word2vec` models based on a large quantity of sentence-level train data, this model has no access to any sentence of this rare language. Therefore, we can not design a model straightly relied on corpus. Fortunately, a `WordNet-style` dictionary is provided. Since we can judge the similarity of any two recorded words via the dictionary, we can design two function f and s, while f encodes a word to a D-dim vector and s evaluates the similarity of two word-vectors.

1.1 The Form of the Vectors

For convince, it is necessary to limit the form of the word-vectors. Because of the continuity and compact of real numbers, we can use vectors within $[0,\ 1]^D$ to represent infinite words theoretically. Therefore, the encoding function f has form:

$$f : word \mapsto v \in [0,\ 1]^D$$

while D is a fixed integer.

In the meantime, s should have this form:

$$s : \left\langle [0,\ 1]^D,\ [0,\ 1]^D \right\rangle \mapsto \mathbb{R}$$

1.2 Pre-train the Vectors

If D is equal to the number of words, we can simply use `one-hot` vectors to represent these words, for we can let words with the same meaning encode in the same vector, and different vectors otherwise. However, it is a waste of computer memory. Usually, D is a not too large number, and we can design a wise encode model to promise that the distance between the vectors of words evaluates the difference of them at most time (a small amount of mistakes is inevitable).

Now, imagine the words are some atoms in a box-shaped space. There is repulsion within some atoms, which means these words have different meanings, meanwhile, there is also attraction representing that some words have similar meanings. Although these atoms may be at random location at first. With the effect of repulsion and attraction, they will finally stop at fixed locations.

After explaining the physical background, we will introduce some pre-set parameters: N for the number of words in total, $r \in [0,\ 1]$ for the ratio of base-point word to all, $L > 0$ for the length to move per step.

First of all, we set all vectors at random loaction within $[0,\ 1]^D$.

Then, in every step, we choose rN vectors as base-points randomly. Based on the effect of the forces of these rN vectors, we will renew the locations of the other $(1 - r)N$ vectors.

Just like algorithm **Simulated Annealing**, to imitate the real physical world can be efficent. Therefore, we choose the form of Newton's gravitation formula to design our renew mechanism. The force from v_1 to v_2:

$$\overrightarrow{F}(v_1,\ v_2) = \frac{\overrightarrow{v_1} - \overrightarrow{v_2}}{\|\overrightarrow{v_1} - \overrightarrow{v_2}\|^2}(-1)^m$$

if `WordNet` shows w_1 and w_2 have different meanings, then $m = 1$, otherwise, $m = 0$.

In step t, we choose series of base-points $\{v_{b_1},\ v_{b_2},\ \cdots,\ v_{b_{rN}}\}$ first, then renew the others

(e.g. renew v_k):

$$v_k^{(t)} = v_k^{(t-1)} + L \cdot \sum_{i=1}^{rN} F(v_{b_i}^{(t-1)},\, v_k^{(t-1)})$$

$$v_k^{(t)} = max\{0,\, min\{1,\, v_k^{(t)}\}\}$$

We recommend that L should be a not too large number, such as 10^{-3} or 10^{-4}.

The parameter r can affect the speed of convergence, when r approaches to 1, it will need a long time to converge, and when r approaches to 0, it may not converge. Therefore, an r between 0.3 and 0.5 may be proper.

If D is too small, the atoms will be bound in a crowded box, so that they can not move by even one micron. If D is too large, it will need more time to converge, and it may be a waste of computer memory sources. Some experience shows that, for millions of words, $D = 300$ works well.

1.3 Similarity Function

We define s with cosine and sigmoid:

$$s(v_1,\, v_2) = \sigma\left(\frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|} + s_0\right)$$

while s_0 is a automatic adjustable parameter, which should keep most $s(v_1,\, v_2)$ reflect the true result (positive for similar, negative for different).

1.4 Unknown Words

Since the `WordNet` dictionary might not contain all words of this rare language, it is necessary to have a certain method to propress the unknown words.

A simple method is to assume unknown words have blended features of all known words, so we set: $v_{unknown} = \frac{1}{N}\sum_{i=1}^{N} v_i$. Sometimes, $v_{unknown}$ might not be an all-zero vector.

2 WSD System ans POS Tagging

In the previous section, we introduce a word-to-vector method, and in this section, we will use these vectors to develop a semi-supervised learning models for WSD and POS.

Before detailing the models, we should stress the help of `GloVe`**(Pennington et al., 2014[1])**. `GloVe` is a reliable model to work on English word embedding, and we can easily have an access to some pre-trained English word embedding result.

With the model in Section 1 and `GloVe`, we can build a translation model, and build WSD and POS based on the translation model.

2.1 Fake Sentence Generator

Because we have no access to any linguistic material to that language, we have no idea about its syntax structure and the grammer. Thus, it is the only thing we can do to gain train data that we should build a sentence gennrator.

Now, we have lots of sentences in English. Via `WordNet`, we can translate English sentences to new-language sentences word-by-word. These fake sentences may have a wrong syntax structure, so we can not use time-series model like LSTM. In this case, CNN pooling can be useful.

Assume an English sentence is $\{e_{i_1},\, e_{i_2},\, \cdots,\, e_{i_k}\}$, we can generate two kinds of fake sentences. True sentence translated word-by-word, $\{d_{i_1},\, d_{i_2},\, \cdots,\, d_{i_k}\}$; and the wrong sentence containing randomly selected k words, $\{d_{j_1},\, d_{j_2},\, \cdots,\, d_{j_k}\}$.

2.2 WSD System

We will detail a three-stage model for WSD, this model is used to judge whether an English sentence and a new-language sentence are have the same meaning or not.

We suppose $\{e_{i_1},\, e_{i_2},\, \cdots,\, e_{i_k}\}$ and $\{d_{x_1},\, d_{x_2},\, \cdots,\, d_{x_k}\}$ (x=*i*/*j*) are given to judge, then the three stages should be as below (e and d are embedded vectors):

Pooling stage:

$$\widetilde{e}_{i_t} = \frac{1}{2h+1}\sum_{y=t-h}^{t+h} e_{i_y}$$

while h is pooling step length, and do the same process to $\{d_x\}$.

Convolution stage:

$$\widetilde{h}_{e,t} = W_e \cdot \widetilde{e}_{i_t}$$

$$\widetilde{h}_{d,t} = W_d \cdot \widetilde{d}_{x_t}$$

while $W_e \in \mathbb{R}^{d \times D_e}$ and $W_d \in \mathbb{R}^{d \times D_d}$ are both trainable matrices.

Decode stage:

$$\widetilde{H}_e = \left[\widetilde{h}_{e,1},\, \widetilde{h}_{e,2},\, \cdots,\, \widetilde{h}_{e,k}\right]$$

$$\widetilde{H}_d = \left[\widetilde{h}_{d,1},\, \widetilde{h}_{d,2},\, \cdots,\, \widetilde{h}_{d,k}\right]$$

$$score = \frac{1}{d} tr\left(\sigma\left(\widetilde{H}_e \cdot \widetilde{H}_d^T\right)\right)$$

while tr for trace, σ for sigmoid. We ecpect $score = 1$ when $x = i$, and $score = 0$ for $x = j$. To train the model, we can use cross entropy to optimize $score$.

After training, we will use the model to build WSD system. When we receive a sentence $\{d_{i_1}, d_{i_2}, \cdots, d_{i_k}\}$, we can translate it to several English sentence word by word via `WordNet`, because of polysemants. Then, we can input these several pairs into the model and get some scores. Finally, we choose the English sentence with the highest score as the WSD result for that given rare-language sentence.

2.3 POS tagging

Compared with WSD introduced above, we also design a three-stage model. In a brief, we do some revisions on the last model. We assume there is P kinds of tags in English, so the model is changed as below:

Pooling: same as WSD

Convolution: fix $d = P$

Decode:

$$\widetilde{H}_e = \left[\widetilde{h}_{e,1}, \widetilde{h}_{e,2}, \cdots, \widetilde{h}_{e,k}\right]$$

$$\widetilde{H}_d = \left[\widetilde{h}_{d,1}, \widetilde{h}_{d,2}, \cdots, \widetilde{h}_{d,k}\right]$$

$$tag = \sigma\left(SVD\left(W_p \cdot \widetilde{H}_e \cdot \widetilde{H}_d^T\right)\right)$$

while SVD for Singular Value Decomposition. In addition, $W_p \in \mathbb{R}^{P \times P}$ is a trainable matrix. Then transfer tag to a one-hot vector, which represents a kind of tagging.

3 Model Explanation

Compared with the traditional methods based on statistics and Bayesian formula and Markov Chain, this model innovate ideas of **word embedding and convolution**.

The **word vectors**, to some extent, can perform better to measure the similarity between the meanings of two words, for a real number is more accurate than a 0-or-1 boolean.

As it is detailed in the WSD and POS model, **convolution** can synthesize more information in the context, especially for the unknown words. When we use Bayesian method, for the unknown words have never appeared in train data, the Bayesian method will do mistakes all the time. However, our model can work in this circumstance.

4 Which one is easier to build?

Comparing WSD model in Section 2.2 and POS model in Section 2.3, it is obvious that WSD model is easier to trained. There is only one layer with $d \times (D_e + D_d)$ parameters to train in WSD, while two layers with $p \times (D_e + D_d) + p \times p$ parameters in POS.

Training WSD will need less data than POS, and the former will be easier and faster to converge.

Meanwhile, the information in `WordNet`(WSD basing on) is more reliable than syntax analysis(POS basing on).

In total, it is easier to build WSD system for this rare language than POS.

5 References

[1] Jeffrey Pennington, Richard Socher, Christopher D. Manning. **GloVe: Global Vectors for Word Representation**. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543.