philipperemy / japanese-words-to-vectors Public

master Go to file Code

philipperemy Merge pull request ... f27549c · 6 years ago

tests	Refactor File/UTF8 h...	8 years ago
.gitignore	Refactor File/UTF8 h...	8 years ago
Dockerfile	Add Dockerfile	8 years ago
README.md	Update README.md	7 years ago
cleanup.sh	Update Cleanup Script	8 years ago
generate_vectors.py	add \n to the functio...	6 years ago
requirements.txt	Update requirement...	7 years ago

README

Japanese Word2Vec

こんにちは！

About

Word2vec (word to vectors) approach for Japanese language using Gensim (Deep Learning skip-gram and CBOW models). The model is trained on the *Japanese version of Wikipedia* available at jawiki-latest-pages-articles.xml.bz2.

Definition: Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a high-dimensional space (typically of several hundred dimensions), with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

Further reading about word2vec:
http://nlp.stanford.edu/projects/glove/

Usage

Generating the vectors from a wikipedia dump takes about 2~3 hours on a Core i5, with the default parameters.

```
git clone https://github.com/philipperemy/japa
cd japanese-word-to-vectors
pip3 install -r requirements.txt # you can crea
wget https://dumps.wikimedia.org/jawiki/latest/

# will use TinySegmenter3 for the tokenization
python3 generate_vectors.py

# recommended. will use the MeCab tokenizer. In
# next section of the README called "Tokenize t
python3 generate_vectors.py --mecab
```

If `generate_vectors.py` does not detect the file `jawiki-latest-pages-articles.xml.bz2` , it will download it automatically before running the long generation of the vectors.

Convert Wiki dump to text

The first step is to extract the text and the sentences of the dump. It is done in this function:

```
INPUT_FILENAME = 'jawiki-latest-pages-articles
JA_WIKI_TEXT_FILENAME = 'jawiki-latest-text.tx
JA_WIKI_SENTENCES_FILENAME = 'jawiki-latest-te
process_wiki_to_text(INPUT_FILENAME, JA_WIKI_TE
```

The output consists of two files:

- `JA_WIKI_TEXT_FILENAME` whose content looks like: `trebuchet msフォントアンバサンドとは意味する記号である` where each line corresponds to an article.
- `JA_WIKI_SENTENCES_FILENAME` where each line corresponds to a sentence or chunk of words in the text. This file will not be used in the word2vec algorithm but can be useful to train a sentence to vec (named skip-thoughts, available here https://github.com/ryankiros/skip-thoughts/).

Tokenize the text

Tokenizing means separating the full text into words by using spaces as delimiters. Two approaches are available here:

TinySegmenter3 (easy but less accurate in the tokenization phase)

For this, we use a library called TinySegmenter3 which is able to tokenize japanese corpus with more than 95 percent accuracy (source: http://lilyx.net/tinysegmenter-in-python/).

The output is `JA_WIKI_TEXT_TOKENS_FILENAME` . It looks like this: `trebuchet ms フォント アンバサンド と は を 意味 する`

MeCab (advanced but very accurate)

I strongly advise you to read this tutorial first: How to install MeCab.

The installation depends on your OS:

MacOS

```
brew install mecab
brew install mecab-ipadic
brew install git curl xz
git clone --depth 1 https://github.com/neologd/
cd mecab-ipadic-neologd
./bin/install-mecab-ipadic-neologd -n
pip3 install mecab-python3
```

Ubuntu

```
sudo apt-get install mecab mecab-ipadic libmeca
sudo apt-get install mecab-ipadic-utf8
sudo apt-get install git curl
git clone --depth 1 https://github.com/neologd/
cd mecab-ipadic-neologd
sudo ./bin/install-mecab-ipadic-neologd -n
pip3 install mecab-python3
```

Infer the vectors

Finally, the Gensim library is used to perform the word2vec algorithm with the parameters:

- size of 50 (dimensionality of the feature vectors)
- window of 5 (maximum distance between the current and predicted word within a sentence)
- min count of 5 (ignore all words with total frequency lower than this)
- iter of 5 (number of iterations or epochs over the corpus)
- number of workers equal to number of cores

While training, the console output looks like:

```
2016-09-04 02:54:38,354 : INFO : PROGRESS: at 1
2016-09-04 02:54:39,346 : INFO : PROGRESS: at 1
2016-09-04 02:54:40,356 : INFO : PROGRESS: at 1
2016-09-04 02:54:41,390 : INFO : PROGRESS: at 1
```

Once it's finished, 4 new files are generated:

- `ja-gensim.50d.data.model` . This file contains the model in the binary format. Use `model = Word2Vec.load(fname)` to get back your word2vec model.

About

Word2vec (word to vectors) approach for Japanese language using Gensim and Mecab.

wikipedia japanese word2vec corpus gensim japanese-language word2vec-algorithm

Readme
Activity
86 stars
3 watching
17 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 3

philipperemy Philippe Rémy
vortexkd Chinnapa
mcps5601 Ying-Jia Lin

Languages

Python 89.2%
Dockerfile 7.7%
Shell 3.1%

- `ja-gensim.50d.data.txt`. This file contains the model vectors in the text format. Can be used in any other script without the Gensim library!
- `ja-gensim.50d.data.model.syn1neg.npy` and `ja-gensim.50d.data.model.wv.syn0.npy`. Files generated automatically. Contains some numpy arrays (weights and other parameters). It must be in the same directory as the model.

Finally, let's inspect `ja-gensim.50d.data.txt`

```
の 0.128774 3.631298 -3.058414 -0.434418 -0.30€
に -2.019490 4.359702 -1.845176 -2.663986 1.774
は 0.296134 4.136690 -3.184480 -0.817397 0.555€
```

Here we can see the vectors for `の`, `に` and `は`. If we go deeper, we can see longer words such as `文献`. The size of the vocabulary is the number of lines of this file (one line equals one word and its vector representation).

`wc -l ja-gensim.50d.data.txt` yields `1200627` words.

References

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their