Sane's STATS Academy of Statistics

Word Embedding Algorithms

gensim library, word2vec

Drawbacks with Vectorizers

- All the words are considered equally irrespective of their synonyms and antonyms
- The meaningfulness in the language cannot be captured

Word Embedding Algorithms

- A modern approach to natural language processing
- Using neural network algorithms, algorithms word2vec and GloVe have been developed
- These algorithms provide a dense vector representation of words that capture something about their meaning
- These algorithms learn about the word by the association the word is used for
- A vector space representation of every word, which given by the embedding algorithms provides a projection where words with similar meanings are locally clustered within the space

Gensim Library

- Gensim is a free Python library designed to automatically extract semantic topics from documents, as efficiently (computer-wise) and painlessly (human-wise) as possible.
- Gensim was developed and is maintained by the Czech natural language processing researcher Radim Řehůřek and his company RaRe Technologies.
- Gensim is designed to process raw, unstructured digital texts ("*plain text*").

Main Features of Gensim

- Gensim includes streamed parallelized implementations of the following:
  - fastText
  - word2vec
  - doc2vec algorithms
  - latent semantic analysis (LSA, LSI, SVD)
  - non-negative matrix factorization (NMF)
  - latent Dirichlet allocation (LDA)
  - tf-idf
  - random projections.

Sane's STATS Academy of Statistics

Word2Vec

Developing Word2Vec Embedding

Word2Vec

- 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 was created by a team of researchers led by Tomas Mikolov at Google and patented.
- There are two main algorithms on which we can train with Word2Vec namely, CBOW (Continuous Bag of Words) and Skip-Grams
- We will be using pre-trained algorithms
- Gensim provides the Word2Vec class for working with a Word2Vec model.

Pre-Trained Examples

- First we load the word2vec model based on Google News data

```
from gensim.models import KeyedVectors
filename = 'GoogleNews-vectors-negative300.bin'
model = KeyedVectors.load_word2vec_format(filename, binary=True)
```

- We can examine the nearness of some words by meaning point of view

```
result = model.similarity('lunch','dinner')
print(result)
```

```
0.68157035
```

```
result = model.similarity('lunch','cook')
print(result)
```

```
0.35912475
```

Sane's STATS Academy of Statistics

Pre-trained Examples

```
result = model.most_similar('lunch',topn=3)
print(result)
```

```
[('lunches', 0.7324951887130737), ('meal', 0.706218838691711 4),
('breakfast', 0.698990941047668 5)]
```

```
result = model.most_similar('Hinduism',topn=3)
print(result)
```

```
[('Sikhism', 0.6996403336524963), ('Jainism', 0.660114645957946 8),
('Buddhism', 0.6583923101425171)]
```

Sane's STATS Academy of Statistics

Pre-trained Examples

```
result = model.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
print(result)
```

```
[('queen', 0.7118192911148071)]
```

```
result = model.most_similar(positive=['woman', 'king'], negative=['man'], topn=5)
print(result)
```

```
[('queen', 0.7118192911148071), ('monarch', 0.6189674139022827),
('princess', 0.5902431011199951), ('crown_prince',
0.5499460697174072), ('prince', 0.5377321243286133)]
```