Bag-of-Words Language Model

Bag-of-words

Bag-of-words(BoW) is a statistical language model used to analyze text and documents based on word count. The model does not account for word order within a document. BoW can be implemented as a Python dictionary with each key set to a word and each value set to the number of times that word appears in a text.

```
# the sentence "Squealing suitcase
squids are not like regular squids."
could be changed into the following BoW
dictionary:

{'squeal': 1, 'like': 1, 'not': 1,
'suitcase': 1, 'be': 1, 'regular': 1,
'squid': 2}
```

Feature Extraction in NLP

Feature extraction (or vectorization) in NLP is the process of turning text into a BoW vector, in which features are unique words and feature values are word counts.

```
# given the following features
dictionary mapping:
{'are':0,
 'many':1,
 'success':2,
 'there':3,
 'to':4,
 'ways':5}

# "many success ways" could be
represented
# as the following BoW vector:
[0, 1, 1, 0, 0, 1]
```

codecademy

Bag-of-words Test Data

Bag-of-words *test data* is the new text that is converted to a BoW vector using a trained *features dictionary*. The new test data can be converted to a BoW vector based on the index mapping of the trained features dictionary.
For example, given the training data "There are many ways to success." and the test data "many success ways", the trained features dictionary and the test data BoW vector could be the following.

```
# the trained features dictionary
{'are':0,
 'many':1,
 'success':2,
 'there':3,
 'to':4,
 'ways':5}

# the test data BoW vector
[0, 1, 1, 0, 0, 1]
```

Feature Vector

In machine learning, a *feature vector* is a numeric depiction of an object's salient features. In the case of bag-of-words (BoW), the objects are text samples and those features are word counts.
For example, given this features dictionary mapping, a BoW feature vector of "Another five fish find another faraway fish." would be $[1, 0, 2, 0, 0, 0, 1, 1, 0, 0, 2]$.

```
{'five': 0,
'fantastic': 1,
'fish': 2,
'fly': 3,
'off': 4,
'to': 5,
'find': 6,
'faraway': 7,
'function': 8,
'maybe': 9,
'another': 10}
```

Language Smoothing in NLP

Language smoothing is a solution to avoid overfitting in NLP. It takes a bit of probability from known words and allots it to unknown words. This causes the unknown words to have a probability of more than 0.

codecademy

Features Dictionary in NLP

A *features dictionary* is a mapping of each unique word in the training data to a unique index. This is used to build out bag-of-words vectors.
For instance, given the training data “Squealing suitcase squids are not like regular squids”, the features dictionary could be as the following.

```
{'squeal': 0, 'suitcase': 1, 'squid': 2,
'be': 3, 'not': 4, 'like': 5, 'regular':
6}
```

Bag-of-words Data Sparcity

Bag-of-words has less data sparsity (i.e., it has more training knowledge to draw from) than other statistical models. When vectorizing a document, the vector is considered sparse if the majority of its values are zero which means that most of the words are not contained in the vocabulary design. BoW also suffers less from overfitting (adapting a model too strongly to training data).

Perplexity in NLP

For text prediction tasks, the ideal language model is one that can predict an unseen test text (gives the highest probability). In this case, the model is said to have lower *perplexity*.
Bag-of-words has higher *perplexity* (it is less predictive of natural language) than other models. For instance, using a Markov chain for text prediction with bag-of-words, you might get a resulting nonsensical sequence like: “we i there your your”.
A trigram model, meanwhile, might generate the far more coherent (though still strange): “i ascribe to his dreams to love beauty”.

Print Share

codecademy