13. feature engineering

Common features used in text classification

~ bag of words

~ N-grams: incorporating more context

A contiguous sequence of n items (characters, tokens) in text.

CPSC330 students are hard-working .

2-grams (bigrams): a contiguous sequence of two words

CPSC330 students, students are, are hard-working, hard-working .

3-grams (trigrams): a contiguous sequence of three words

CPSC330 students are, students are hard-working, are hard-working .

spaCy: A useful package for text processing and feature extraction

If we want to go beyond bag-of-words and incorporate human knowledge in models, we carry out feature engineering.

Some common features include:

~ ngram features

~ part-of-speech features

~ named entity features (spaCy)

~ emoticons in text

These are usually extracted from pre-trained models using libraries such as spaCy.

Now a lot of this has moved to deep learning.

But industries still rely on manual feature engineering.

Feature selection:

Find the features (columns) X that are important for predicting y, and remove the features that aren’t.

**Feature selection can often result in better performing (less overfit), easier to understand, and faster model.**

13. feature selection

1. logistics regression is easy to interpret, svm is not but can still benefit from feature selections (SelectFromModel)

2. RFE (Recursive feature elimination)

~ Build a series of models. At each iteration, discard the least important feature according to the model.

~ computationally expensive (slow)

Basic idea

1. fit model

2. find least important feature

3. remove

4. iterate.

Detailed steps:

1. Decide k, the number of features to select.

2. Assign importances to features, e.g. by fitting a model and **looking at coef\_ or feature\_importances**\_.

3. Remove the least important feature.

4. Repeat steps 2-3 until only k features are remaining.

How do we know what value to pass to n\_features\_to\_select?

Use RFECV which uses cross-validation to select number of features.

3. Search and score

Example: Suppose you have three features: A,B,C

Compute score for S={}

Compute score for S={A}

Compute score for S={B}

Compute score for S={C}

Compute score for S={A,B}

Compute score for S={A,C}

Compute score for S={B,C}

Compute score for S={A,B,C}

Return S with the best score.

How many distinct combinations we have to try out?

Forward or backward selection

Shrink or grow feature set by removing or adding one feature at a time

Makes the decision based on whether adding/removing the feature **improves the CV score** or not

Text, letter

Description automatically generated

Difference between this forward/backward selection and RFE is the former’s decision based on CV score and the latter based on coef\_ or feature\_importance