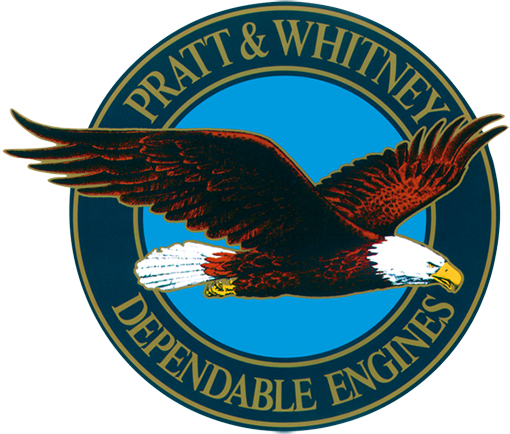
Mahmoud Kojok

United Technologies Corporation

Last Edited: 2019-04-26



Keyword extraction

# Architecture:

Test texts

Training texts

Candidate keywords identification

Candidate keywords identification

Feature generation

Feature calculation

Keyword ranking

Learning

Model

# The Algorithm:

This is a **supervised** keyword extraction algorithm, unlike the Rake method (included in the SOW Datascience folder > keywords > Rake method for keyword extraction) which is an unsupervised keyword extraction method.

The algorithm should have two stages:

1. Training: create a model for identifying keywords, using training short texts/an excel file containing numerous previous texts under a single column where the keywords are known.
2. Extraction: choose keywords from a new short text, using the above model.

Both stages choose a set of candidate keywords from their input texts, and then calculate the values of certain features for each candidate.

We describe these two steps first, and then outline the training and extraction stages in more detail.

# Candidate keywords

The candidate keywords are chosen in the following steps:

It first cleans the input text, then identifies candidates, and finally stems and case-folds the phrases.

***Data Preprocessing***

The input text is split into tokens (sequences of letters, digits and internal periods), and then several modifications are made:

• punctuation marks, brackets, and numbers are replaced

• apostrophes are removed;

• hyphenated words are split in two;

***Keyword identification***

1. Candidate keywords cannot be proper names (Words that start with a capital letter only).
2. Candidate keywords cannot be a stopword.

The stopword list contains the most used words that are not keywords and that should be filtered, such as (the, that, this) or just single letters. Most stopword lists can be found online, or can be created manually.

*Case-folding and stemming*

The final step in determining candidate keywords is to case-fold all words and stem them using the iterated Lovins method.

Stemming and case-folding allow us to treat different variations on a keyword as the same thing.

# Feature calculation

Two features are calculated for each candidate keyword and used in training and extraction. They are:

TF×IDF, a measure of a phrase’s frequency in a text compared to its rarity in general use; and first occurrence, which is the distance into the text of the keywords’ first appearance.

TF×IDF:

The TF×IDF for keyword P in text D is:

Where:

1. freq(P,D) is the number of times P occurs in D
2. size(D) is the number of words in D

First occurrence

The second feature, first occurrence, is calculated as the number of words that precede the Keywords’ first appearance, divided by the number of words in the text. The result is a number between 0 and 1 that represents how much of the text precedes the keywords’ first appearance.

# Training: building the model

The training stage uses a set of training texts for which the keywords are known (these are provided by CYIENT and can be found in the SOW Datascience folder (keywords > provided by cyient). For each training text, candidate keywords are identified and their feature values are calculated as described above. Each word is then marked as a keyword or a non keywords, using the actual keywords for that text. This binary feature is the class feature used by the machine learning scheme.

The scheme then generates a model that predicts the class using the values of the other two features.

*Machine learning scheme:*

The Naïve Bayes technique because it is simple and yields good results. This scheme learns two sets of numeric weights from the discretized feature values, one set applying to positive (“keyword”) examples and the other to negative (“not a keyword”) instances.

# Extraction of new keywords

To select keywords from a new text, candidate keywords are determined then the model built during training is applied. The model determines the overall probability that each candidate is a keyword. When the Naïve Bayes model is used on a candidate word with feature values t (for TF×IDF) and d (for distance), two quantities are computed:

and a similar expression for P[no], where Y is the number of positive instances in the

training texts, and N is the number of negative instances, that is, when candidate keywords are not keywords. (The Laplace estimator is used to avoid zero probabilities. This simply replaces Y and N by Y+1 and N+1.)

The overall probability that the candidate phrase is a keyphrase can then be calculated:

Candidate words are ranked according to this value.

# Correcting misspelled keywords:

Two different ways for correcting misspelled keywords have been found, the first is the Levenshtein distance which measures the distance between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions, or substitutions) required to change one word into the other. You can read more about this online, a sample levensthein distance algorithm has also been included in the SOW Datascience folder > keywords > extractor with spelling corrector > Levenshtein edit distance.

The second way which is explained online (<https://norvig.com/spell-correct.html>) uses edits1 and edits2 that look into candidate correct words for a given misspelled keyword.

# POS tagging:

Keywords with negation before them such as (no, not) are not taken as candidate keywords and should be classified as non-keywords. Example: “no bearing failure”, in this case, even if bearing was found to be a keyword using our algorithm, it should be classified a non-keyword and disregarded simply because there is “no” before it.

POS tagging could help with this task, but other methods can be used.

More on tagging words: <https://www.nltk.org/book/ch05.html>