KEYWORDS DETECTION BY USING PYTHON LIBRARY :

**[Contents](https://github.com/MaartenGr/KeyBERT" \l "table-of-contents)**

1. [About the Project](https://github.com/MaartenGr/KeyBERT#about)  
   . [Installation](https://github.com/MaartenGr/KeyBERT#installation)  
    [Max Sum Distance](https://github.com/MaartenGr/KeyBERT#maxsum)  
    [Maximal Marginal Relevance](https://github.com/MaartenGr/KeyBERT#maximal)  
    [Embedding Models](https://github.com/MaartenGr/KeyBERT#embeddings)
2. [Large Language Models](https://github.com/MaartenGr/KeyBERT#llms)

[**About the Project**](https://github.com/MaartenGr/KeyBERT#about) **:**

Keyword detection, often referred as keyword spotting. It is a crucial task in various applications, including natural language processing (NLP).

Some of the open source LIBRARIES and AI tools used for keyword detection are:

1. NLTK (Natural Learning Toolkit)
2. Spacy
3. Gensim
4. Rapid Automatic Keyword Extraction

Machine Learning Models used for Keyword Detection are:

1. Bidirectional Encoder Representations from Transforms
2. Term Frequency – Inverse document frequency

Here I’m going to use BERT model for Keyword Detection.

**KeyBert model:**

First, document embeddings are extracted with BERT to get a document-level representation. Then, word embeddings are extracted for N-gram words/phrases. Finally, we use cosine similarity to find the words/phrases that are the most similar to the document. The most similar words could then be identified as the words that best describe the entire document.

**Installation :**

pip install keybert is used for installation.

We can install more libraries depending upon transformers and language backends that we are using.

pip install keybert[flair]

pip install keybert[gensim]

pip install keybert[spacy]

pip install keybert[use]

* **Spacy** is a python library for natural language processing and it can be used to extract keywords and phrases from text.
* **Gensim** is used for topic modelling and document similarity analysis .
* **Rapid Automatic Keyword Extraction (RAKE)** is an open source algorithm designed for keyword extraction. It’s simple and efficient.

**CODE FOR KEYWORD DETECTION :**

text = """ Supervised learning is the machine learning task of learning a function that

maps an input to an output based on example input-output pairs. It infers a

function from labeled training data consisting of a set of training examples.

In supervised learning, each example is a pair consisting of an input object

(typically a vector) and a desired output value (also called the supervisory signal).

A supervised learning algorithm analyzes the training data and produces an inferred function,

which can be used for mapping new examples. An optimal scenario will allow for the

algorithm to correctly determine the class labels for unseen instances. This requires

the learning algorithm to generalize from the training data to unseen situations in a

'reasonable' way (see inductive bias)

.

The KeyBERT library makes keyword extraction easy and efficient.

"""

# Initialize the KeyBERT model

kw\_model = KeyBERT('distilbert-base-nli-mean-tokens')

# Extract keywords

keywords = kw\_model.extract\_keywords(text, keyphrase\_ngram\_range=(1, 2), stop\_words='english', top\_n=5)

# Print the keywords

for keyword in keywords:

print(keyword[0]) # The keyword

1. We import KeyBert class from the keybert libraray.
2. Now, we provide a sample text that we want to extract keywords from.
3. We install the KeyBert model with pre-trained BERT model.
4. We use extract\_keywords method to extract keywords from text.
5. The keyphrase\_ngram\_range parameter defines the length of extracted phrases and top n parameters specifies how many top keywords to extract.
6. Now, print the extracted keywords.

We can highlight the keywords in the document by simply setting highlight:

keywords = kw\_model.extract\_keywords(doc, highlight=True)

### [Max Sum Distance](https://github.com/MaartenGr/KeyBERT" \l "23-max-sum-distance)

The maximum sum distance between pairs of data is defined as the pairs of data for which the distance between them is maximized.

To diversify the results, we take the 2 x top\_n most similar words/phrases to the document. Then, we take all top\_n combinations from the 2 x top\_n words and extract the combination that are the least similar to each other by cosine similarity.

kw\_model.extract\_keywords(doc, keyphrase\_ngram\_range=(3, 3), stop\_words='english',

use\_maxsum=True, nr\_candidates=20, top\_n=5)

[('set training examples', 0.7504),

('generalize training data', 0.7727),

('requires learning algorithm', 0.5050),

('supervised learning algorithm', 0.3779),

('learning machine learning', 0.2891)]

Maximal Marginal Relevance

1. MMR considers the similarity of keywords/keyphrases with the document, along with the similarity of already selected keywords and keyphrases.
2. Depending upon how we set the number of candidates, diversity can be either high or low.
3. Maximal Marginal Relevence tries to maximize the diversity and minimize the redundancy.
4. It is always better to prefer high diversity or maximum sum to get the key phrases.

kw\_model.extract\_keywords(doc, keyphrase\_ngram\_range=(3, 3), stop\_words='english',

use\_mmr=True, diversity=0.7)

[('algorithm generalize training', 0.7727),

('labels unseen instances', 0.1649),

('new examples optimal', 0.4185),

('determine class labels', 0.4774),

('supervised learning algorithm', 0.7502)]

Pros and Cons in Keyword Detection:

* Keyword detection models, also known as wake word or trigger word detection models, have several advantages and disadvantages. These models are commonly used in voice-controlled systems and smart devices to trigger specific actions or commands. Here are some pros and cons of keyword detection models:

Pros:

1. **Efficiency:** Keyword detection models are designed to operate in an always-on mode, consuming very little power. They are energy-efficient and can run on low-power devices.

**2. Privacy:** These models are often designed to operate locally on the device without the need for continuous cloud connectivity.

3. **Quick Response:** When a keyword is detected, it can quickly activate the device or system, leading to rapid response times for user commands or actions.

**4**. **Customization:** Users can often customize the keyword to suit their preferences, making it more natural and personal.

Cons:

1. **Limited Vocabulary**: Keyword detection models are constrained by a specific keyword or set of keywords. They cannot handle open-domain conversations or understand a wide range of commands.

2. **Sensitivity to Noise:** They can be sensitive to background noise, which might trigger false positives or make it difficult for the model to recognize the keyword.

3**. Lack of Context**: Keyword models lack context awareness. They only detect keywords and do not understand the context of the user's command or conversation.

4. **Limited Use Cases**: These models are primarily useful for specific applications like voice assistants, and their utility is limited to tasks that involve triggering actions based on a keyword.

**Accuracy:**

1. **Data Annotation**: Assuming you have a sizable dataset for training and validation, with 100,000 minutes of data, you can create a labled dataset where you manually tag examples of keywords and non-keywords.

This step is critical for training your model accurately.

2. **Model Choice**: Choose a well-suited pre-trained NLP model and fine-tune it on your labeled data. Fine-tuning is crucial to adapt the model to your specific keyword detection task.

1. **Evaluation:**  After training, evaluate your model's performance. Given the large dataset, you can expect relatively high accuracy, the exact accuracy will depend on the complexity of the keywords and the quality of your annotation.

**Costs**

1. **Data Preparation:** Preparing 100,000 minutes of data for training and validation can

be time-consuming and might require data cleaning. In this process costs will

include labor and possibly data storage.

1. **Model Training**: The cost of model training depends on factors like the model's architecture, batch size, and computational resources.
2. **Data Labelling**: If you need human annotation for data labelling, it can be costly.

Labelling 100,000 minutes of data is a significant task. Consider outsourcing or using

a data labelling service, which will have associated costs.

1. **Maintenance and Updates**: Models require periodic updates to stay accurate. Budget

for ongoing maintenance, which may include monitoring, retraining, and updating.

1. **Infrastructure**: Costs for maintaining infrastructure to host your model in a production environment, including cloud hosting fees, can be significant.
2. **Scalability**: Plan for potential scalability costs as your data volume and prediction

requirements grow.

1. **Software Development**: If you need custom software development for model integration, monitoring, or maintenance, factor in the associated development

costs.

It's important to note that these projections are based on several assumptions and simplifications. The actual accuracy and costs can vary significantly based on the complexity of your keyword detection task.