1. Scan the directories to select doc, docx and pdf file
2. Open file and extract the text
3. Extract the features from the file
4. Feature

References

1. <https://medium.com/@winston.smith.spb/python-ocr-for-pdf-or-compare-textract-pytesseract-and-pyocr-acb19122f38c>
2. <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>
3. <https://xiaofeima1990.github.io/2016/12/19/extract-text-from-sanned-pdf/>
4. <http://spie.org/newsroom/3819-analysis-and-classification-for-complex-scanned-documents?SSO=1>
5. <https://automatetheboringstuff.com/chapter13/>
6. <https://media.readthedocs.org/pdf/pdfminer-docs/latest/pdfminer-docs.pdf>

<http://okfnlabs.org/blog/2016/04/19/pdf-tools-extract-text-and-data-from-pdfs.html>

<https://github.com/mgorkove/pdfToTxt/blob/master/code/pToT.py>

pdfminer: <https://stackoverflow.com/questions/40665067/extract-pdf-using-pdfminer-with-whitespace>

<https://medium.com/@rqaiserr/how-to-convert-pdfs-into-searchable-key-words-with-python-85aab86c544f>

<http://zacstewart.com/2015/04/28/document-classification-with-scikit-learn.html>

Feature extraction:

http://scikit-learn.org/stable/modules/feature\_extraction.html#tfidf-term-weighting

1. **Research objectives**

The main objective is to automatically collect scientific documents in the ER network in order to save time and money by either avoiding a second purchase or simply avoiding losing them. It corresponds to the development of a machine-learning classification solution to classify scientific documents from non-scientific ones.

1. **Data collection**

This step has been done.

Number of sci docs: 122

1. **Data preparation**
2. Reading the data

There are two kinds of documents:

* Text documents: The documents are read as text

This is done

* Scanned documents: The documents are scanned, they contain text but are recognized as images

1. Extract the features of the document:

The features will be textual or graphical features.

* Text document

Extract the text from the document, and count the words that will be used to identify the type of document. They are:

keywords, abstract, volume, vol., volume+number, no., by, rev., journal, jnl, laboratory, university, univ., institute, research doi.org, published, received, original article

mot-clés, resumé, par

* Scanned documents: images

1. **Data exploration**
2. **Model selection**

**Support Vector Machines, Random Forest, Gradient Boosting Machines**

1. **Presentation**

**Documents**

**Preprocessing + Feature Extraction**

**Training a SVM Classifier**

**Features**

**Labels**

**Training phase**

**Prediction phase**

**Document**

**Trained Classifier**

**Label**

**Preprocessing + Feature Extraction**

**Features**

Figure . Document classification based on machine learning methods

Documents

===========Index = 8==================

D:\Work\TextAnalysis\DocClassification\SelectedDocs\Deep\_Convolutional\_Neural\_Networks\_for\_Hyperspectr.pdf

article 1

{'keywords': 1, 'par': 0, 'no.': 0, 'univ.': 0, 'authors': 1, 'rev.': 0, 'by': 6, 'resumé': 0, 'received': 2, 'jnl': 0, 'abstract': 1, 'institute': 0, 'doi': 1, 'journal': 0, 'research': 1, 'publication': 1, 'mot-cles': 0, 'introduction': 1, 'tutorial': 0, 'index': 1, 'paper': 2, 'published': 2, 'laboratory': 3, 'volume+number': 0, 'university': 1, 'conference': 0, 'article': 1}

The words are attached together

**For the refined data**

Model with rank: 1

Mean validation score: 0.880 (std: 0.048)

Parameters: {'C': 1, 'kernel': 'linear'}

Model with rank: 2

Mean validation score: 0.873 (std: 0.049)

Parameters: {'C': 10, 'kernel': 'linear'}

Model with rank: 3

Mean validation score: 0.867 (std: 0.012)

Parameters: {'C': 10, 'kernel': 'poly'}

Model with rank: 3

Mean validation score: 0.867 (std: 0.023)

Parameters: {'C': 100, 'kernel': 'poly'}

Model with rank: 1

Mean validation score: 0.886 (std: 0.061)

Parameters: {'n\_estimators': 50, 'max\_features': 2}

Model with rank: 2

Mean validation score: 0.880 (std: 0.053)

Parameters: {'n\_estimators': 1000, 'max\_features': 2}

Model with rank: 3

Mean validation score: 0.867 (std: 0.066)

Parameters: {'n\_estimators': 100, 'max\_features': 2}

Model with rank: 3

Mean validation score: 0.867 (std: 0.066)

Parameters: {'n\_estimators': 1000, 'max\_features': 10}

Model with rank: 3

Mean validation score: 0.867 (std: 0.045)

Parameters: {'bootstrap': False, 'n\_estimators': 10, 'max\_features': 10}

Model with rank: 1

Mean validation score: 0.905 (std: 0.059)

Parameters: {'n\_estimators': 1000, 'learning\_rate': 0.2, 'max\_features': 15}

Model with rank: 2

Mean validation score: 0.892 (std: 0.050)

Parameters: {'n\_estimators': 100, 'learning\_rate': 0.5, 'max\_features': 15}

Model with rank: 3

Mean validation score: 0.886 (std: 0.054)

Parameters: {'n\_estimators': 100, 'learning\_rate': 0.1, 'max\_features': 15}