Automatically categorize Artifacts

How do we prepare the artifacts for machine learning?

Text

Libraries: NLTK, spaCy

Tokenize document

Part-of-speech tagger

Remove stopwords

Words are lemmatized and stemmed

Use bag of words with **TF-IDF** → Normalize word count (TF) and assign relevance to words based on corpus count (IDF)

Pictures

Tesseract and OpenCV

Open source

Really bad performance without prior image processing

Highly adaptable, can be trained with handwritten letters

No third party API's needed, everything possible programmatically

Google OCR API

Needs business account and costs

JSON returned with text, confidence and position of bounding boxes

Really good performance for printed text, average for handwritten

Not really adaptable

Where do we get the categories (classes) from?

Algorithms

Latent Dirichlet Analysis (LDA) - `gensim.models.ldamulticore` or mallets lda

Latent Semantic Analysis (LSA). Uses Singular Value Decomposition (SVD) on the Document-Term Matrix. Can be used for feature selection) – `genism.models.lsimodel`

Non-negative Matrix Factorization (NMF) – `sklearn.decomposition.NMF`

Text Rank for text summarization

K-Means together with **Principal Component Analysis** (PCA) - `sklearn.decomposition.PCA` + `sklearn.cluster.KMeans`

Gaussian mixture models - `sklearn.mixture.GaussianMixture`

Challenges

How do we get class labels from clusters?

On text data you can extract most important words

Use Crowd sourcing to interprete cluster

What is the best cluster size?

Hierarchical Dirichlet process

Try out which is the best cluster for k = 1…n

Evaluation

How good is a cluster?

Elements in cluster similar to representative

Elements in different clusters dissimilar

Silhouette coefficient

Adjusted Mutual Information Score

Topic coherence score

`sklearn.metrics`

How do we categorize new artifacts?

Algorithms

Naïve Bayes - `sklearn.naive_bayes.MultinomialNB`

Support vector machine (SVM) - `sklearn.linear_model.SGDClassifier`

Neuronal Networks – `TensorFlow`

Bagging - Train classifiers on partitions on data and let them vote

`sklearn.esemble.BaggingClassifier`

Boosting - Classifiers are trained in sequence on misclassified data.
`sklearn.esemble.AdaBoostClassifier`

Evaluation

variance = $E[(m(x) - E(m(x))^2]$ classifier is overly optimized to training data

High variance → model random noise → overfitting

Overfitted models perform poorly on test data

bias = $E[(m(x) - f(x)]$ = Error from wrong assumptions

High bias → miss relevant information → underfitting

Classification accuracy

```
np.mean(predicted == test_data.target)
```

Precition, Recall, F1-Score

```
classification_report(test_data.target, predicted, target_names=data.target_names)
```

Training time / Prediction time

Robustness (Tendency for overfitting)

Interpretability

Scalability / Compactness

Split labeled data into training set and a test set

`sklearn.model_selection.train_test_split`

Test set will be used to evaluate your classifier

m-fold cross validation

Split data into m subsets and train m times with m-1 parts training data and 1 set validation data

Combine the m generated models to an overall model
`sklearn.model_selection.GridSearchCV`