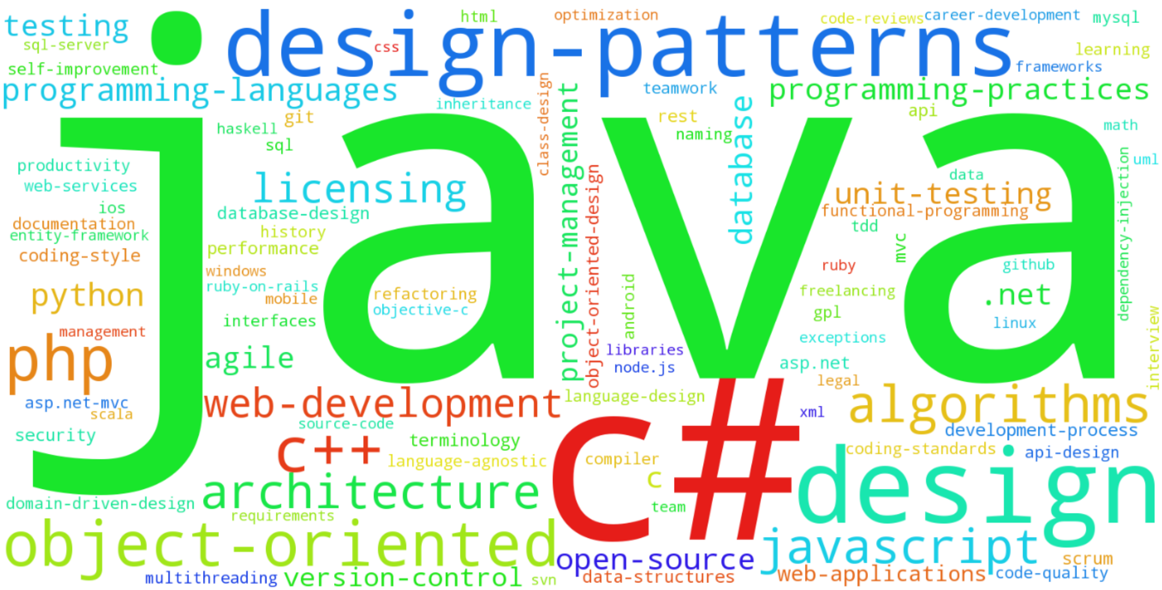
Project 4: Machine Learning

Team #7: Michael Herold, Ralph Samer



Recap

Task

- Automatic tagging of unseen posts using both supervised & unsupervised approach

Data set

- StackExchange Data Dump from March, '16
- Subgroup: programmers.stackexchange.com
- Training set: 34,500 posts (~90%)
- Test set: 3,800 posts (~10%)

StackExchange

Total number of …	
Posts	38,315
Tags	1,614

Tags per post	
Min	1
Max	5
Average	2.68

Current Approach

Preprocessing

Posts

- Tokenized content
  - *Body*
  - *Title*
  - *Accepted answer (if not negative score)*
- Ignored posts
  - *Score < threshold and no accepted answer*
  - *No answers and negative score*

- Removed HTML, code, URLs, emoticons, numbers, single letters
- Removed stop words
- Stemming *(nltk.stem.porter.PorterStemmer)*
- Merge synonym words

Tags

- Merge similar tags
- Remove less frequent tags

Current Approach

Features & Transformation

Feature Selection:

- word occurences as features
  - remove words having little impact on performance of our models
- Try additional features that improve performance
  (e.g. whether tag appears in title or body of a document/post or not)

Challenges:

- highly depends on preprocessing (stop-word removal, merge synonyms, ...)
- crucial for performance (of models)
- when words are features:
  - important words should appear frequently
  - but not in every single document/post

Goal:
get best prediction accuracy

Transformation:

- e.g. TF-IDF, less frequent words occuring more often
  in one post are more important for that post than others $TFxIDF_{ij} = TF_{ij} log(\frac{N}{DF_j})$

Current Approach

Supervised (I)

Start to train a Multinomial Naive Bayes classifier (*MultinomialNB from sklearn)*

Two Naive Bayes approaches:

- First approach:
  - Train multiple *binary* classifiers (1 classifier/tag)
  - classes = {true, false}
  - Apply classifier on test set => suggest k most probable tags

- Second approach:
  - Train a single classifier <u>for all</u> *n* tags
  - #classes = n
  - Apply classifier on test set => suggest k most probable tags

Reasons for choosing classification over (linear) regression:

- We have n Tags: n is a discrete/natural number
- Also: predicting if a post contains a specific tag is discrete/binary problem
- ➢ Predicting tags is a typical classification problem
  - tags act as classes (= output of classifiers)

Current Approach

Supervised (II)

First tests with Naïve Bayes (full data set):

- sufficient results, but still enough space left for improvements (e.g. improve preprocessing, feature selection, weight important features, …)

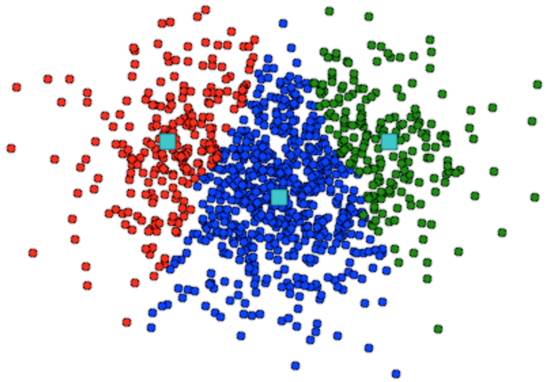
Compare performance of Naïve Bayes with other supervised models:

- **k-Nearest Neighbors:**
  - suggest h most probable tags
  - probability depends on:
    - number of neighbors that contain these suggested tags
    - distance to these neighboring documents/posts
- **Linear SVM:**
  - Use only a few features that represent the entire post

Current Approach

Unsupervised

- **Approach:**
  - Cluster similar posts according to the *TFxIDF* matrix
  - Assign *n* most frequent tags of cluster to new samples of cluster
- **k-Means clustering:**
  - Number of clusters (k): #tags
  - Initialization: *k-means++*
  - Distance: *Euclidean*



- **Hierarchical Agglomerative Clustering:**
  - Linkage: *Ward criterion* (minimum variance)
  - Distance: *Euclidean*
  - Stop criterion: number of clusters (#tags)

Evaluation

- $$\text{Precision} = \frac{true\_positives}{true\_positives + false\_positives}$$
- $$\text{Recall} = \frac{true\_positives}{true\_positives + false\_negatives}$$

Both should be high → **F1 measure**

- True positives: if tag found in prediction set and original set
- False positives: if tag found in prediction set but not in original set
- False negatives: if tag found in original set but not in prediction set

Current Approach

First results

Model	Precision	Recall
MultinomialNB	0.332	0.285
k-Means [1)]	0.298	0.262
HAC [2)]	0.281	0.255

All results are based on <u>two predicted tags</u> (average tag assignment = 2.68)

[1)] Number of clusters = #tags

[2)] Stop criterion = #clusters → #tags

Problems

Preprocessing:

- High dimensionality → sparse matrix
- POS-tagging and lemmatization not working good enough
- Merge synonyms and similar tags

Learning:

- Choosing proper number of clusters (*k*) → dendrogram from HAC
- Finding best fitting model and #tags to predict

Runtime & memory:

- Too many features for forward/backward selection
- HAC very intensive to compute

Expected results & Outlook

Expected results:

- F1 not ideal but tags are reasonable
- Precision decreases by the number of suggested tags
- Recall increases by the number of suggested tags

Outlook:

- Improve preprocessing
- Use different (more independent) features for Naïves Bayes
- Evaluation

Thank you for your attention.

References:

- *Knowledge Discovery and Data Mining 1* by Denis Helic and Roman Kern (2015)
- Segaran, T. 2007. *Programming Collective Intelligence: Building Smart Web 2.0 Applications. S*ebastopol: O'Really