Text Technologies for Data Science
INFR11145

Text Classification

Instructor: Walid Magdy

31-Oct-2018

Lecture Objectives

• **Learn** about text basic of text classification
  • Definition
  • Types
  • Methods
  • Evaluation
Text Classification

- **Text classification** is the process of classifying documents into predefined categories based on their content.

  - Input: Text (document, article, sentence)
  - Task: Classify into predefined one/multiple categories
  - Categories:
    - Binary: relevant/irrelevant, spam .. etc.
    - Few: sports/politics/comedy/technology
    - Hierarchical: patents

Classification is and is not

- **Classification** (a.k.a. “categorization”): a ubiquitous enabling technology in data science; studied within pattern recognition, statistics, and machine learning.

- Definition:
  the activity of predicting to which among a predefined finite set of groups (“classes”, or “categories”) a data item belongs to

- Formulated as the task of generating a hypothesis (or “classifier”, or “model”)

  \[ h : D \rightarrow C \]

  where \( D = \{x_1, x_2, \ldots\} \) is a domain of data items and \( C = \{c_1, \ldots, c_n\} \) is a finite set of classes (the classification scheme)
Classification is and is not

- Different from clustering, where the groups (“clusters”) and their number are not known in advance.
- The membership of a data item into a class must not be determinable with certainty.
  - e.g., predicting whether a natural number belongs to Prime or NonPrime is not classification.
- In text classification, data items are
  - Textual: e.g., news articles, emails, sentences, queries, etc.
  - Partly textual: e.g., Web pages.

Types of Classification

- Binary:
  - item to be classified into one of two classes
  - \( h : D \rightarrow C, \quad C = \{c_1, c_2\} \)
  - e.g., Spam/not spam, male/female, rel/irrel
- Single-Label Multi-Class (SLMC)
  - item to be classified into only one of \( n \) possible classes.
  - \( h : D \rightarrow C, \quad C = \{c_1, \ldots, c_n\}, \text{ where } n>2 \)
  - e.g., Sports/politics/entertainment, positive/negative/neutral
- Multi-Label Multi-Class (MLMC)
  - item to be classified into none, one, two, or more classes
  - \( h : D \rightarrow 2^C, \quad C = \{c_1, \ldots, c_n\}, \text{ where } n>1 \)
  - e.g., Assigning CS articles to classes in the ACM Classification System
  - Usually be solved as \( n \) independent binary classification problems.
Dimension of Classification

- Text classification may be performed according to several dimensions ("axes") orthogonal to each other
- By topic; by far the most frequent case, its applications are global
- By sentiment; useful in market research, online reputation management, social science and political science
- By language (a.k.a. "language identification"); useful, e.g., in query processing within search engines
- By genre; e.g., AutomotiveNews vs. AutomotiveBlogs, useful in website classification and others;
- By author (a.k.a. "authorship attribution"), by native language ("native language identification"), or by gender; useful in forensics and cybersecurity
- By usefulness; e.g., product reviews
- ……

Rule-based classification

- An old-fashioned way to build text classifiers was via knowledge engineering, i.e., manually building classification rules
  - E.g., (Viagra or Sildenafil or Cialis) → Spam
  - E.g. (#MAGA or America great again) → support Trump
- Common type: dictionary-based classification
- Disadvantages:
  - Expensive to setup and to maintain
  - Depends on few keywords → bad coverage (recall)
Supervised-learning classification

- A generic (task-independent) learning algorithm is used to train a classifier from a set of manually classified examples
- The classifier learns, from these training examples, the characteristics a new text should have in order to be assigned to class $c$
- Advantages:
  - Generating training examples cheaper than writing classification rules
  - Easy update to changing conditions (e.g., addition of new classes, deletion of existing classes, shifted meaning of existing classes, etc.)
Extract Features

- In order to be input to a learning algorithm (or a classifier), all training (or unlabeled) documents are converted into vectors in a common vector space
- The dimensions of the vector space are called features
- In order to generate a vector-based representation for a set of documents $D$, the following steps need to be taken
  1. Feature Extraction
  2. Feature Selection or Feature Synthesis (optional)
  3. Feature Weighting

Step 1: Feature Extraction

- What are the features that should be different from one class to another?
- Simplest form: BOW
  - Each term in a document is a feature
  - Feature space size = vocabulary in all docs
  - Standard IR preprocessing steps are usually applied
    - Tokenisation, stopping, stemming
- Other simple features forms:
  - Word n-grams (bigrams, trigrams, ....)
    - Much larger + more sparse
  - Sometimes char n-grams are used
    - Especially for degraded text (OCR or ASR outputs)
Step 1: Feature Extraction

• What other text features could be used?

• Sentence structure (NLP):
  • POS (part-of-speech tags)
  • Syntactic tree structure

• Topic-based features (NLP):
  • LDA topics
  • NEs (named entities) in text
  • Links / Linked terms

• Non-textual features:
  • Average doc\sentence\word length
  • % of words start with upper-case letter
  • % of links/hashtags/emojis in text

Step 1: Feature Extraction

• What preprocessing to apply?
  • Case-folding? really vs Really vs REALLY
  • Punctuations? “?”, “!”, “@”, “#”
  • Stopping? “he”, “she”, “what”, “but”
  • Stemming? “replaced” vs “replacement”

• Other Features:
  • Start with Cap, All Cap
  • Repeated characters “congraaaaats” “help!!!!!!!!!”
  • LIWC: Linguistic Inquiry and Word Count

• Which to choose?
  • Classification task/application
Step 2: Feature Selection

- Number of distinctive features = feature space = length of feature vector.
- Vector can be of length $O(10^6)$, and might be sparse → High computational cost → Overfitting
- What are the most important features among those?
  - e.g. Reduce $O(10^6)$ to $O(10^4)$
- For each class, find the top representative $k$ features for it → get the Union over all classes → reduced feature space

Step 2: Feature Selection Functions

- Document frequency
  - % of docs in class $c_i$ that contain the term $t_k$
  - Very basic measure. Will select stop words as features
    $$\#(t_k, c_i) = P(t_k | c_i)$$
- Mutual Information
  - How term $t_k$ appear in class $c_i$ compared to other classes
  - Highly used in feature selection in text classification
    $$MI(t_k, c_i) = \sum_{c \in \{c_i, \bar{c}_i\}} \sum_{t \in \{t_k, \bar{t}_k\}} P(t, c) \cdot \log_2 \frac{P(t, c)}{P(t) \cdot P(c)}$$
- Pearson’s Chi-squared ($x^2$)
  - used more in comparisons between classes
Step 2: Feature Selection Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Denoted by</th>
<th>Mathematical form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document frequency</td>
<td>$(t_k,c_i)$</td>
<td>$P(t_k</td>
</tr>
<tr>
<td>DIA association factor</td>
<td>$z(t_k,c_i)$</td>
<td>$P(c_i</td>
</tr>
<tr>
<td>Information gain</td>
<td>$IG(t_k,c_i)$</td>
<td>$\sum_{c_1 \in</td>
</tr>
<tr>
<td>Mutual information</td>
<td>$MI(t_k,c_i)$</td>
<td>$\log \frac{P(t_k,c_i)}{P(t_k) \cdot P(c_i)}$</td>
</tr>
<tr>
<td>Chi-square</td>
<td>$\chi^2(t_k,c_i)$</td>
<td>$\frac{[\text{Tr} \cdot (P(t_k,c_i) \cdot P(\tilde{t}_k,\tilde{c}_i) - P(t_k,\tilde{c}_i) \cdot P(\tilde{t}_k,c_i))]^2}{P(t_k) \cdot P(\tilde{t}_k) \cdot P(c_i) \cdot P(\tilde{c}_i)}$</td>
</tr>
<tr>
<td>NCL coefficient</td>
<td>$NCL(t_k,c_i)$</td>
<td>$\sqrt{\text{Tr} \cdot [P(t_k,c_i) \cdot P(\tilde{t}_k,\tilde{c}_i) - P(t_k,\tilde{c}_i) \cdot P(\tilde{t}_k,c_i)]}$</td>
</tr>
<tr>
<td>Relevancy score</td>
<td>$RS(t_k,c_i)$</td>
<td>$\log \frac{P(t_k</td>
</tr>
<tr>
<td>Odds Ratio</td>
<td>$OR(t_k,c_i)$</td>
<td>$\frac{P(t_k</td>
</tr>
<tr>
<td>GSS coefficient</td>
<td>$GSS(t_k,c_i)$</td>
<td>$P(t_k,c_i) \cdot P(\tilde{t}_k,\tilde{c}_i) - P(t_k,\tilde{c}_i) \cdot P(\tilde{t}_k,c_i)$</td>
</tr>
</tbody>
</table>

Step 2: Feature Synthesis

- **Matrix decomposition techniques** (e.g., PCA, SVD, LSA) can be used to synthesize new features that replace the features discussed above.

- These techniques are based on the principles of **distributional semantics**, which states that the semantics of a word “is” the words it co-occurs with in corpora of language use.
  - **Pros**: the synthetic features in the new vectorial representation do not suffer from problems such as polysemy and synonymy.
  - **Cons**: computationally expensive.

- **Word embeddings**: the “new wave of distributional semantics”, as from “deep learning”.
  - PCA: Principle component analysis
  - SVD: Singular value decomposition
  - LSA: latent semantic analysis.
Step 3: Feature Weighting

- Attributing a value to feature $t_k$ in document $d_i$
  
  This value may be
  
  - binary (representing presence/absence of $t_k$ in $d_i$);
  - numeric (representing the importance of $t_k$ for $d_i$);

  obtained via feature weighting functions in the following two classes:
  
  - unsupervised: e.g., tfidf or BM25,
  - supervised: e.g., $tf \times MI$, $tf \times x^2$

- The similarity between two vectors may be computed via cosine similarity; if these vectors are pre-normalized, this is equivalent to computing the dot product between them

Training a Classifier

- For binary classification, essentially any supervised learning algorithm can be used for training a classifier;
  
  popular choices include
  
  - Support vector machines (SVMs)
  - Random forests
  - Naïve Bayesian methods
  - Lazy learning methods (e.g., k-NN)
  - …

- The “No-free-lunch principle” (Wolpert, 1996) \(\rightarrow\) there is no learning algorithm that can outperform all others in all contexts

- Implementations need to cater for
  
  - the very high dimensionality
  - the sparse nature of the representations involved
Training a Classifier

- For **Multiclass classification**, some learning algorithms for binary classification are “SLMC-ready”; e.g.
  - Decision trees
  - Random forests
  - Naive Bayesian methods
  - Lazy learning methods (e.g., k-NN)

- For other learners (notably: SVMs) to be used for SLMC classification, combinations / cascades of the binary versions need to be used
  - e.g. multi-class classification SVM
  - Could be directly used for MLMC as well

Parameter Optimisation of Classifier

- Most classifiers has some parameters to be optimized:
  - The $C$ parameter in soft-margin SVMs
  - The $r, d$ parameters of non-linear kernels
  - Decision threshold for binary SVM

- Optimising the parameters on test data is cheating!

- Data Split:
  - Usually labelled data would be split into **three parts**
    - **Training**: used to train the classifier (typically 80% of the data)
    - **Validation**: used to optimise parameters. Apply the classifier on this data with different values of the parameters and report the one that achieves the highest results (usually 10% of the data)
    - **Test**: used to test the performance of the trained classifier with the optimal parameters on these unseen data (usually 10% of the data)
Cross-Validation

- Sometimes the amount of labelled data in hand is limited (e.g. 200 samples). Having evaluation of a set of 20 samples only might be misleading
- Cross-validation is used to train the classifier with all data and test on all data without being cheating
- Idea:
  - Split the labelled data into \( n \) folds
  - Train classifier on \( n-1 \) fold and test on the remaining one
  - Repeat \( n \) times
- 5-fold cross validation
- Extreme case: LOOCV
  LOOCV: leave-one-out cross-validation

Evaluation

- Efficiency / Effectiveness
- Baselines
- Efficiency:
  - Speed in learning
    - SVM with linear kernel is known to be fast
    - DNNs are known to be much slower (specially with large # layers)
  - Speed in classification
    - K-NNs are known to be one of the slowest
  - Speed in feature extraction
    - BOW vs POS vs Link analysis features
- Effectiveness:
  - Global effectiveness measures
  - Per class effectiveness measures
Evaluation: Baselines

- There are standard methods for creating baselines in text classification to compare your classifier with.

- Most popular/simplest baselines
  - Random classification
    - Classes are assigned randomly
    - How better classifier is doing than random?
  - Majority class baseline
    - Assign all elements to the class that appears the most
    - How better you are doing that the stupidest classifier
  - Simple algorithm, e.g. BOW
    - Usually used when you introduce new interesting features

Evaluation: Binary Classification

- Accuracy:
  - How many of the samples are classified correctly?
  - \( A = \frac{9}{10} = 0.9 \)
Evaluation: Binary Classification

- A = 7/10 = 0.7  
- A = 7/10 = 0.7
- When classes are highly unbalanced
  - Precision/recall/F1 for the rare class
  - e.g. Spam classification (detection)

### Evaluation: Binary Classification

<table>
<thead>
<tr>
<th></th>
<th>System 1</th>
<th>System 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>1/3 = 0.33</td>
<td>0/1 = 0</td>
</tr>
<tr>
<td>Recall</td>
<td>1/2 = 0.5</td>
<td>0/2 = 0</td>
</tr>
<tr>
<td>F1</td>
<td>0.4</td>
<td>0</td>
</tr>
</tbody>
</table>
**Evaluation: Multi-class**

- Majority class baseline
- Accuracy = 0.8
- Macro-F1 = 0.296

- Macro-F1:
  - Should be used in binary classification when two classes are important
  - e.g.: males/females while distribution is 80/20%

---

**Error Analysis**

- **Confusion Matrix**
  How classes get confused?

<table>
<thead>
<tr>
<th></th>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C₂</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C₃</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- Useful:
  - Find classes that get confused with others
  - Develop better features to solve the problem
Summary

- Text Classification tasks
- Feature extraction/selection/synthesis/weighting
- Learning algorithms
- Cross-validation
- Baselines
- Evaluation measures
  - Accuracy/precision/recall/Macro-F1

Resources

- Fabrizio Sebastiani
  Machine Learning in Automated Text Categorization
  ACM Computing Surveys, 2002
  Link: https://arxiv.org/pdf/cs/0110053