Lecture 4
Text classification

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
10.03.2021
Plan for today

- Text classification tasks
- Text representations for classification
- Generative vs discriminative models
- Logistic regression classifier
- Text classification with feed-forward neural networks
- Evaluating text classifiers
Text classification tasks
Text classification
Text/document classification

- Ads based on user posts (topic)
- classify posts based on depression symptoms (anything clinical)
- Emails (spam vs non-spam), different categories
- news (category), detecting false news
- Language classification
- Sentiment analysis
- chatbot, intent detection
- customer service complaints
- personal properties (age, gender etc)
Text/document classification

- Spam detection
- Topic classification (sport/finance/travel etc)
- Genre classification (news/sports/fiction/social media etc)
- Sentiment analysis
- Emotion prediction
- Dialogue intent detection
Authorship attribution

- Detecting the author of a particular text
- Native language identification
- Clinical text classification
  - diagnosing psychiatric or cognitive impairments
  - Predicting diagnosis from clinical notes
- Identification of gender, dialect, educational background, level of language proficiency etc
Types of classification tasks

- Binary classification (true/false, 1/0, 1/-1)
- Multi-class classification (text genre: politics/finance/travel)
- Multi-label classification (image captioning)
- Clustering: mostly unsupervised
  - Topic modeling – important but will not talk about it today
Multi-label classification

- What if each document can have multiple labels?
- If the labels are **flat**, like when predicting movie genres based on plot descriptions

![Comedy](image1)
- Comedy

![Comedy, Crime, Thriller](image2)
- Comedy, Crime, Thriller

![Drama](image3)
- Drama

![Animation, Comedy, Family](image4)
- Animation, Comedy, Family

![Comedy, Romance](image5)
- Comedy, Romance
If the labels are flat ...

- Turn the multi-label classification problem into several binary classification problems
- Form “super-classes” from label combinations and perform multi-class classification
If the labels are hierarchical …

- Like for instance in categorizing articles based on some ontology.
If labels are hierarchical

- We can still build a flat classifier
  - Each path in the ontology is one class
  - chemistry|organic chemistry|steroids

- We can still build binary classifiers for each label but it makes less sense
  - because we know that we should predict organic chemistry only after we have predicted chemistry

- Build hierarchical models
  - lots of small models, one for each node
    - model that is used only after chemistry has been predicted and that predicts only labels inorganic chemistry and organic chemistry
  - One model that makes serial predictions
    - $P(\text{chemistry}|\text{doc}) \times P(\text{organic chemistry}|\text{doc, chemistry}) \times P(\text{peptides}|\text{doc, chemistry, organic chemistry})$
Classification vs regression

- What if the labels are ordered?
- For instance, sentiment scores: -3, -2, -1, 0, 1, 2, 3
  - -3 – the sentiment/attitude is extremely negative
  - 0 – the sentiment/attitude is neutral
  - 3 – the sentiment/attitude is very positive
- With multi-class classification all scores are similarly dissimilar
  - Here 2 and 3 are more similar than -3 and 3
- It might be good idea to try regression instead of classification to predict the continuous score
Text representation for classification
How to represent a document?

- Bag of Words (BOW)
- Hand-crafted features
- Learned feature representations
BOW representations – based on word counts

- Binary BOW

<table>
<thead>
<tr>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Multinomial BOW

<table>
<thead>
<tr>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Tf-idf
Tf-idf

- **Tf-idf** = **tf** (term frequency) x **idf** (inverse document frequency)

- **Term frequency:**
  - Raw frequency
  - Binary
  - Raw frequency normalized by the document length

- **Inverse document frequency**
  - **N** - the number of documents
  - **nt** – the number of documents containing term **t**

\[ \text{idf} = \log \frac{N}{n_t} \]
Tf-idf example

- $\text{Tf-idf}(\text{doc4, the}) = 0$
- $\text{Tf-idf}(\text{doc4, your}) = 0$
- $\text{Tf-idf}(\text{doc4, model}) = 0$
- $\text{Tf-idf}(\text{doc4, cash}) = 1 \log \frac{5}{3}$
- $\text{Tf-idf}(\text{doc4, Viagra}) = 3 \log 5$
- $\text{Tf-idf}(\text{doc4, class}) = 0$
- $\text{Tf-idf}(\text{doc4, account}) = \log \frac{5}{4}$
- $\text{Tf-idf}(\text{doc4, orderz}) = \log \frac{5}{2}$

\[
tf\text{-}idf = n_{d,t} \cdot \log \frac{N}{n_t}
\]
Tf-idf example

- Tf-idf(doc4, the) = 0
- Tf-idf(doc4, your) = 0
- Tf-idf(doc4, model) = 0
- Tf-idf(doc4, cash) = log(5/3) = 0.51
- Tf-idf(doc4, Viagra) = 3log5 = 4.83
- Tf-idf(doc4, class) = 0
- Tf-idf(doc4, account) = log(5/4) = 0.22
- Tf-idf(doc4, orderz) = log(5/2) = 0.92

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Tf-idf example

tf-idf = \textit{n}_{d,t} \cdot \log \frac{N}{n_t}

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>0</td>
<td>0</td>
<td>1.61</td>
<td>0</td>
<td>0</td>
<td>1.83</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.04</td>
<td>0</td>
<td>0</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>doc3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.92</td>
<td>0.22</td>
<td>0</td>
</tr>
<tr>
<td><strong>doc4</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
<td><strong>0.51</strong></td>
<td><strong>4.83</strong></td>
<td><strong>0</strong></td>
<td><strong>0.22</strong></td>
<td><strong>0.92</strong></td>
</tr>
<tr>
<td>doc5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.02</td>
<td>0</td>
<td>0</td>
<td>0.22</td>
<td>0.92</td>
</tr>
</tbody>
</table>
Preprocessing for BOW or tf-idf representation

- Tokenization
- Stemming/Lemmatization
- Remove stop words
- Remove infrequent words
- Replace certain word tokens with special entities
  - Replace all numeric tokens with NUM
- Lowercase the text
- Social media - handle emoticons

- Always consider your task to decide what kind of preprocessing makes sense
Variations of BOW – word n-grams

new price update for the cake decoration tools from Alick

new price update for the cake decoration tools from Alick
Variations of BOW – character n-grams

el gato se sentó en la alfombra
BOW representations in scikit-learn

- sklearn.feature_extraction.text.CountVectorizer
- sklearn.feature_extraction.text.TfidfVectorizer
How to represent a document?

- Bag of Words (BOW)
  - PRO: Easy, no effort required
  - CON: Ignores sentence structure

- Hand-crafted features

- Learned feature representations
Generative vs discriminative classifiers
Logistic regression
Generative task formulation

Given document $d$ and a set of class labels $C$, assign to $d$ the most probable label $\hat{c}$.

Bayes rule

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)$$

Naïve Bayes
Discriminative task formulation

- Given document $d$ and a set of class labels $C$, assign to $d$ the most probable label $\hat{c}$.

$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

Logistic regression
Generative vs discriminative models

- **Generative (joint) models – P(c, d)**
  - Model the probability of both input document and output label
  - Can be used to generate a new document with a particular label
  - N-gram models, Naïve Bayes

- **Discriminative (conditional) models - P(c|d)**
  - Learn boundaries between classes. Input data is taken as given.
  - Logistic regression, SVM, most neural models
Logistic regression

- Like linear regression but for classification

- Input is a feature vector $x = [x_1, x_2, \ldots, x_m]$

- In binary case:

  $$P(c = 1|x; \mathbf{w}) = \frac{1}{1 + e^{-x \cdot \mathbf{w}}}$$

- Trained with binary cross-entropy loss
Logistic regression – multiple classes

- Instead of a parameter vector, there is a parameter matrix
- One column for every class, e.g. one parameter vector for every class

\[ z = x^T W \]

- Instead of sigmoid, there is softmax

\[ p(c|x; W) = \text{softmax}(z) \]

- This combo – a linear layer + sigmoid/softmax – is identical to what is often used in neural networks output.
Features in a logistic regression model

- Can use BOW and tf-idf
- But can also use more complex linguistic features
- For instance, for sentiment classification:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>count(positive lexicon) ∈ $doc$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>count(negative lexicon) ∈ $doc$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\begin{cases} 1 \text{ if } \text{&quot;no&quot; } \in doc \ 0 \text{ otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>count(1st and 2nd pronouns) ∈ $doc$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$\begin{cases} 1 \text{ if } \text{&quot;!&quot; } \in doc \ 0 \text{ otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>log(word count of $doc$)</td>
</tr>
</tbody>
</table>
How to represent a document?

- **Bag of Words (BOW)**
  - **PRO:** Easy, no effort required
  - **CON:** Ignores sentence structure

- **Hand-crafted features**
  - **PRO:** Can use NLP resources and pipelines, class-specific features
  - **CON:** Relies on (language-specific) linguistic resources, need to do feature engineering

- Learned feature representations
Text classification with feed-forward neural networks
Feed-forward sentence classification

\[ h = g(W^{(i)} x + b^{(i)}) \]

\[ z = W^{(o)} h + b^{(o)} \]

\[ p(c|x) = \text{softmax}(z) \]

Embedding layer can be initialized with the pretrained word embeddings.
FastText text classification

- Very quick and simple text classification baseline, often with good performance
- Joulin et al., 2017. Bag of Tricks for Efficient Text Classification
- C++ code available: https://github.com/facebookresearch/fastText
- Trains the sub-word-based word embeddings along with the classifier
- Can also use pre-trained FastText embeddings
FastText text classification

- It is essentially a CBOW model
- Form a text representation by averaging the embeddings of all words in text

\[
\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3 \quad \mathbf{v}_4 \quad \mathbf{v}_5 \quad \mathbf{v}_6 \quad \mathbf{v}_7 \quad \mathbf{v}_8 \quad \mathbf{v}_9 \quad \mathbf{v}_{10}
\]

\[
h = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_i \quad \quad p(c|\mathbf{h}) = \text{softmax}(\mathbf{hW}) \quad \quad W \in \mathbb{R}^{d \times |C|}
\]
How to represent a document?

- **Bag of Words (BOW)**
  - PRO: Easy, no effort required
  - CON: Ignores sentence structure

- **Hand-crafted features**
  - PRO: Can use NLP resources and pipelines, class-specific features
  - CON: Relies on (language-specific) linguistic resources, need to do feature engineering

- **Learned feature representations**
  - PRO: Can learn to represent task-relevant information
  - CON: Need to be learned
Default baselines for text classification

- Logistic regression or SVM for binary classification
- Logistic regression for multiclass classification
- Fasttext classification model
- When the dataset is small, it also makes sense to try Naive Bayes

- Logistic regression, SVM and Naive Bayes are available in sklearn
Evaluating text classification
Evaluation measures

- **Accuracy**: not suitable when the classes are unbalanced
- **Precision**: also called positive predictive value
- **Recall**: also called true positive rate or sensitivity
- **F1-score**: combines precision and recall
Accuracy

- Accuracy = \frac{\#Correct}{\#Total}

<table>
<thead>
<tr>
<th></th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold Predicted</td>
<td>Y Y N N Y N N Y N N N</td>
</tr>
<tr>
<td>Predicted</td>
<td>N Y N Y N N N Y N N N</td>
</tr>
</tbody>
</table>

- Correct = 
- Total = 
- Accuracy =
Accuracy

\[ \text{Accuracy} = \frac{\text{Correct}}{\text{Total}} \]

<table>
<thead>
<tr>
<th></th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>Y          Y          N    N    Y          N    N    N    Y          N    N    N</td>
</tr>
<tr>
<td>Predicted</td>
<td>N        Y          N    Y          N    N    N    N    Y          N    N    N</td>
</tr>
</tbody>
</table>

- Correct = 8
- Total = 11
- Accuracy = 72.7%
Accuracy – another example

- Accuracy = \frac{\#Correct}{\#Total}

<table>
<thead>
<tr>
<th></th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>N N N N Y N N N N N N N</td>
</tr>
<tr>
<td>Predicted</td>
<td>N N N N N N N N N N N N</td>
</tr>
</tbody>
</table>

- Correct =
- Total =
- Accuracy =
Accuracy – another example

- Accuracy = \( \frac{\#Correct}{\#Total} \)

<table>
<thead>
<tr>
<th></th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>N N N N N Y N N N N N N</td>
</tr>
<tr>
<td>Predicted</td>
<td>N N N N N N N N N N N</td>
</tr>
</tbody>
</table>

- Correct = 10
- Total = 11
- Accuracy = 90.9%
Confusion matrix

Actual

<table>
<thead>
<tr>
<th></th>
<th>class 1</th>
<th>class 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>True positives</td>
<td></td>
</tr>
<tr>
<td>FP</td>
<td>False positives</td>
<td></td>
</tr>
<tr>
<td>FN</td>
<td>False negatives</td>
<td></td>
</tr>
<tr>
<td>TN</td>
<td>True negatives</td>
<td></td>
</tr>
</tbody>
</table>

Predicted

<table>
<thead>
<tr>
<th></th>
<th>class 1</th>
<th>class 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>class 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>class 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Precision, recall and F1-score

- TP – true positives – correctly predicted positives
- FP – false positives – negatives that were predicted as positives
- FN – false negatives – positives that were predicted as negatives

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}
\]

\[
F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]
Precision and recall

Precision = \frac{TP}{TP + FP}

Recall = \frac{TP}{TP + FN}

Where:
- \( TP \) = True Positives
- \( FP \) = False Positives
- \( FN \) = False Negatives

Precision: How many selected items are relevant?
Recall: How many relevant items are selected?
## Precision and recall: example

<table>
<thead>
<tr>
<th></th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>Y          Y          N  N  Y      N  N  N  Y      N  N  N</td>
</tr>
<tr>
<td>Predicted</td>
<td>N          Y          N  Y      N  N  N  N  Y      N  N  N</td>
</tr>
</tbody>
</table>

- True Positives =
- False Positives =
- False Negatives =

- Precision =
- Recall =
- F1-score =
## Precision and recall: example

<table>
<thead>
<tr>
<th>Gold</th>
<th>Predicted</th>
<th>Document about Sports?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

- True Positives = 2
- False Positives = 1
- False Negatives = 2

- Precision = 2/3
- Recall = 2/4
- F1-score = 4/7
Precision, recall and F1-score in multi-class setting

- Precision, recall and F1-score assume binary classification

- In the multi-class setting, for each class c:
  - treat the class c as the positive class
  - treat all other classes as negative classes
  - Now can compute precision, recall and F1-score with respect to every class

- Combine the class-specific scores using either micro-averaging or macro-averaging
Micro- and macro-averaging

Micro-averaging
- Count TP, FP and FNs for all classes and sum them up
- Compute the precision, recall and F1-score
- Takes class imbalances more into account

Macro-averaging
- Compute precision, recall and F1-score with respect to every class
- Average the scores
- The scores for each class make an equal contribution

Weighted macro-averaging
- Weight each class score according by the relative number of instances of that class