Lecture 2

Text classification

LTAT.01.001 – Natural Language Processing
Kairit Sirts (kairit.sirts@ut.ee)
15.02.2022
Reading test 1

- Average score 3.16 (min: 1.0, max 4.0)
- Average time: 16 minutes (min: 7.7, max: 20)
- Average difficulty: 3.1,
Plan for today

● Text classification tasks
● Text representations for classification
● Generative models: Naive Bayes
● Discriminative models: Logistic regression
● Evaluating text classifiers
Text classification tasks
Text classification

- Technology
- Sports
- Entertainment
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 (keywords of an article)
- 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.jpg)
- Comedy

![Comedy](image2.jpg)
- Comedy

![Drama](image3.jpg)
- Drama

![Animation](image4.jpg)
- Animation
  - Comedy
  - Family

![Comedy](image5.jpg)
- Comedy
- Crime
- Thriller

![Comedy](image6.jpg)
- 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

- comedy
- comedy, crime, thriller
- drama
- animation, comedy, family
- comedy, romance
If the labels are hierarchical ... 

- Like for instance when 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
    - if **chemistry** has been predicted, then use a model 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
  - In case of regression, 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 word?

- Word is a symbolic variable.
- We need to convert it to a numeric value.
- Solution: represent it as a **one-hot** vector.

<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>the</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>your</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>model</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>cash</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Viagra</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>class</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>account</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>orderz</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
How to represent a document?

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

- 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>

- 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>

- 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
  - In log-scale

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

$$idf = \log \frac{N}{n_t}$$
Tf-idf example

- $Tf-idf(doc4, \text{the}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{your}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{model}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{cash}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{Viagra}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{class}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{account}) = \quad \quad \quad \quad $  
- $Tf-idf(doc4, \text{orderz}) = \quad \quad \quad \quad $ 

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

<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>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><strong>doc4</strong></td>
<td><strong>14</strong></td>
<td><strong>2</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>3</strong></td>
<td><strong>0</strong></td>
<td><strong>1</strong></td>
<td><strong>1</strong></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(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) = $3\log 5 = 4.83$
- Tf-idf(doc4, class) = 0
- Tf-idf(doc4, account) = $\log(5/4) = 0.22$
- Tf-idf(doc4, orderz) = $\log(5/2) = 0.92$

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

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

<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
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 (topic of the next lecture)**
Generative vs discriminative classifiers
Naive Bayes and Logistic regression
Generative task formulation

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

\[
\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)
\]
Naive Bayes: a generative classifier

- Naive Bayes is one of the simplest text classifiers
- It uses the BOW representation of text
- That means:
  - it knows the words the document consists of (and maybe also the frequencies of those words)
  - it does not know the ordering of the words
- Historically, Naive Bayes was a good choice in case of small datasets
- Nowadays, Naive Bayes is not so relevant anymore but it is still good to know it.
Naive Bayes

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

- \( P(c) \) – prior probability of the classes (before seeing the document \( d \) under consideration)

- \( P(d|c) = P(w_1, \ldots, w_n|c) \) – the probability of the document after knowing the fact that it as the class \( c \).
Naive Bayes: the likelihood term

\[ P(w_1, ..., w_n | c) \]

- **Naive Bayes assumption**: let’s assume that the words in the document are independent (given the class)

\[ P(w_1, ..., w_n | c) = P(w_1 | c) \cdot ... \cdot P(w_n | c) \]

- This means that we can use BOW representation, the ordering of the words in not relevant anymore

- P.S: Conditional independence != independence

\[ P(w_1 | c) \cdot ... \cdot P(w_n | c) \neq P(w_1) \cdot ... \cdot P(w_n) \]
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
    - More precisely: can generate the document representation used by a particular model
  - Because of this generative process, it’s not easy to add more complex features
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

- 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) $\in$ doc</td>
</tr>
<tr>
<td>$x_2$</td>
<td>count(negative lexicon) $\in$ doc</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\begin{cases} 1 \text{ if &quot;no&quot; } &amp; \in \text{ doc} \ 0 \text{ otherwise} \end{cases}$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>count(1st and 2nd pronouns) $\in$ doc</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$\begin{cases} 1 \text{ if &quot;!&quot; } &amp; \in \text{ doc} \ 0 \text{ otherwise} \end{cases}$</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 (topic of the next lecture)
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
    - More precisely: can generate the document representation used by a particular model
  - Because of this generative process, it’s not easy to add more complex features

- Discriminative (conditional) models - $P(c|d)$
  - Learn boundaries between classes. Input data is taken as given.
  - Logistic regression, SVM, most neural models
Default baselines for text classification

- Regularised logistic regression or SVM for binary classification
- Regularised 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
Hyperparameter tuning for logistic regression

- Hyperparameter tuning can be very important
- Usually L2-regularisation is good
- Tuning the regularisation parameter is usually the most important hyperparameter
- Use grid search, that contain couple of values from several orders of magnitude
- For instance:
  - 0.01  0.02  0.05
  - 0.1  0.2  0.5
  - 1    2    5
Principles of training, developing and testing the models

- General principles:
  - Train on the train part
  - Develop the model by evaluating on the dev part
  - Test on the test part

- Don’t develop on the test part!
- The test part should be put away (behind several locks) until you have the final model you are happy with.
- Ideally, the test part should be used only once.
- If you need to test a generalization during model development
  - use cross-validation on the training set for developing
  - test on the validation set
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>Gold</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

- Correct =
- Total =
- Accuracy =
Accuracy

- Accuracy = \frac{\#Correct}{\#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</td>
</tr>
<tr>
<td>Predicted</td>
<td>N Y N Y N N N N Y N N</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Gold</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

• Correct = 
• Total = 
• Accuracy =
Accuracy – another example

• Accuracy = \( \frac{\text{Correct}}{\text{Total}} \)

<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 = 10
• Total = 11
• Accuracy = 90.9%

• This is majority class baseline
Confusion matrix

- True positives (TP)
- False positives (FP)
- False negatives (FN)
- True negatives (TN)
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

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

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

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

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

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

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</td>
</tr>
<tr>
<td>Predicted</td>
<td>N Y N Y N N N N Y N N</td>
</tr>
</tbody>
</table>

- True Positives =
- False Positives =
- False Negatives =
- Precision =
- Recall =
- F1-score =
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</td>
</tr>
<tr>
<td>Predicted</td>
<td>N Y N Y N N N Y N 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, macro-averaging, or weighted 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