

Lecture 6: Evaluation and NLP Features

First-Year Project 4:
Natural Language Processing

Christian Hardmeier

5 May 2022

IT UNIVERSITY OF COPENHAGEN

Plans

- ▶ Today: *Evaluation, Data augmentation, Features for NLP*
- ▶ Next week (Week 19): **No on-site lectures**
 - ▶ *Classifiers*: Recordings will be available on LearnIT.
 - ▶ TA sessions will be held as usual.
 - ▶ Group supervision for projects:
Office hours on **Thu 19 May** (regular lecture hours)
 - ▶ Use the forum for any questions!
- ▶ **Fri 3 June**: Project submission

IT UNIVERSITY OF COPENHAGEN

Perplexity about perplexity

- ▶ Definition of perplexity:

$$PPL = p(w_1, w_2, \dots, w_N)^{-\frac{1}{N}} = \frac{1}{\sqrt[N]{p(w_1, w_2, \dots, w_N)}}$$



- ▶ This is undefined whenever $p(w_1, w_2, \dots, w_N) = 0$.
- ▶ A maximum-likelihood will estimate a probability of 0 if there is any unknown word in the sentence.
- ▶ Unknown words must always be expected!
- ▶ This is one of the reasons why we need smoothing!

IT UNIVERSITY OF COPENHAGEN

Evaluating Classifiers

IT UNIVERSITY OF COPENHAGEN

Confusion matrix

		Predicted labels										
		ADJ	ADP	ADV	CONJ	DET	NOUN	PRON	PROPN	PUNCT	VERB	X
Gold labels	ADJ	1385	4	63	0	0	243	0	0	0	88	1
	ADP	4	1734	15	2	0	15	0	0	0	3	248
	ADV	62	97	906	0	29	71	0	0	0	10	27
	CONJ	1	387	71	761	12	3	0	0	0	3	8
	DET	9	3	3	1	2228	17	11	0	0	1	0
	NOUN	109	3	6	0	20	3965	1	0	2	85	5
	PRON	23	30	6	0	204	155	1779	0	0	21	1
	PROPN	35	0	2	1	4	1811	2	0	0	19	5
	PUNCT	37	9	8	1	5	263	0	0	2783	45	7
	VERB	51	10	8	0	2	243	0	0	0	3954	4
	X	36	15	192	1	18	386	20	0	4	120	436

Accuracy

- Accuracy is the proportion of elements classified correctly.

$$\text{Accuracy} = \frac{\text{sum of diagonal}}{\text{total sum}}$$



- Accuracy is the simplest metric for classification.
- It can be misleading in unbalanced datasets!
- It provides no information about individual classes.


IT UNIVERSITY OF COPENHAGEN

Precision and Recall


- ▶ Class-specific metrics:
Based on *single rows and columns* of confusion matrix.
- ▶ Common metrics in NLP come from *Information Retrieval*.
- ▶ In a database search, we want to find
 - ▶ all relevant results, and
 - ▶ no distracting irrelevant results.

IT UNIVERSITY OF COPENHAGEN

Precision and Recall

- ▶ **Precision:**
Out of the examples we **predicted to be** in a certain class, how many of them are correct?
(How many irrelevant results did we find?) 

$$\text{Precision} = \frac{\text{single diagonal element}}{\text{sum of a single column}}$$

- ▶ **Recall:**
Out of the examples **that actually belong** to a certain class, how many of them did we find?
(Did we actually find what we were looking for?) 

$$\text{Recall} = \frac{\text{single diagonal element}}{\text{sum of a single row}}$$

IT UNIVERSITY OF COPENHAGEN

Confusion matrix

		Predicted labels						
		NOUN	PRON	PROPN	PUNCT	VERB	X	
Gold labels		243	0	0	0	88	1	
		15	0	0	0	3	248	
		71	0	0	0	10	27	
		3	0	0	0	3	8	
		17	11	0	0	1	0	
		3965	1	0	2	85	5	
	NOUN	109	3	6	0	20		
	PRON	23	30	6	0	204		
	PROPN	35	0	2	1	4		
	PUNCT	37	9	8	1	5		
	VERB	51	10	8	0	2		
	X	36	15	192	1	18		

$$\text{Precision} = \frac{\text{diagonal element}}{\text{column sum}}$$

$$= \frac{3965}{7172} = 0.553$$

Confusion matrix

	Predicted labels						
	NOUN	PRON	PROPN	PUNCT	VERB	X	
Gold labels		243	0	0	0	88	1
		15	0	0	0	3	248
		71	0	0	0	10	27
		3	0	0	0	3	8
		17	11	0	0	1	0
		109	3	6	0	20	3965
		23	30	6	0	204	155
NOUN	109	3	6	0	20	3965	1
PRON	23	30	6	0	204	155	1779
PROPN	35	0	2	1	4	1811	2
PUNCT	37	9	8	1	5	263	0
VERB	51	10	8	0	2	243	0
X	36	15	192	1	18	386	20

$$\text{Recall} = \frac{\text{diagonal element}}{\text{row sum}}$$

$$= \frac{3965}{4196} = 0.945$$

Precision/Recall vs. Sensitivity/Specificity

- Precision and Recall focus on the true positives in the context of *what was found* and *what should have been found*.
- Sensitivity and Specificity focus on *correct identification of positives and negatives*.
- Sensitivity is just another name for Recall, but *Specificity and Precision are different*.
- Se and Sp are like “positive and negative Recall”.

$$P = \frac{TP}{TP + FP} \quad R = \text{Se} = \frac{TP}{TP + FN} \quad \text{Sp} = \frac{TN}{TN + FP}$$

IT UNIVERSITY OF COPENHAGEN

Precision and Recall can be gamed!

- **100% Precision** (good chance):
Return only the one example you're most certain of!
- **100% Recall** (guaranteed):
Return the entire dataset.
- But *you can't game both of them at the same time!*

IT UNIVERSITY OF COPENHAGEN

F-score (or F-measure)

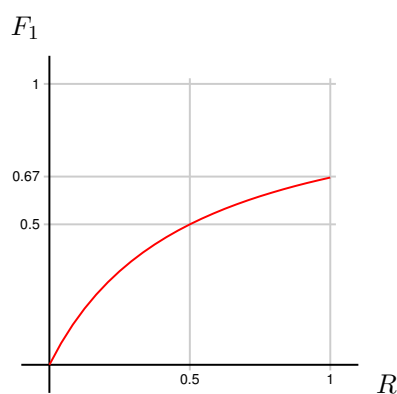
- ▶ We often want to summarise Precision and Recall in a single number.
- ▶ **F-score** (or F_1) is the *harmonic mean* of Precision and Recall.

$$F = 2 \cdot \frac{P \cdot R}{P + R}$$

- ▶ If P and R are equal, then F is the same.
- ▶ If they are different, then F is closer to the *lower* of them.
- ▶ By maximising F-score, we emphasise balanced P and R.
- ▶ F-score can be generalised with a parameter to control the balance.

IT UNIVERSITY OF COPENHAGEN

F-score behaviour



Precision fixed at 0.5

IT UNIVERSITY OF COPENHAGEN

Micro-averaging

- ▶ Precision and Recall are per class, but sometimes we'd like to have *one single number* to characterise our performance.
- ▶ Accuracy is a single number, but is problematic with unbalanced data.
- ▶ **Micro-averaged Precision, Recall and F-score:**
Add the counts of all classes, then compute Precision, Recall and F-score.
- ▶ If each example has only one label, this is *the same as accuracy*.

IT UNIVERSITY OF COPENHAGEN

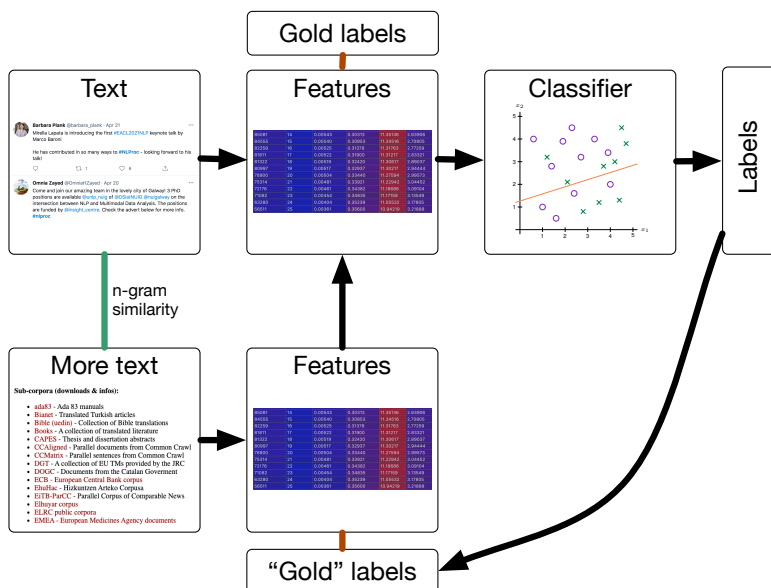
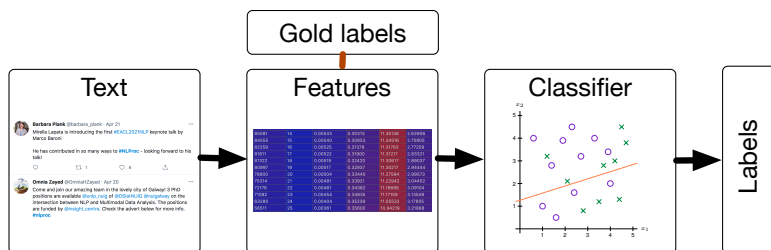
Macro-averaged scores

- **Macro-averaged Precision, Recall and F-score:**
Compute P, R and F for each class,
then take the arithmetic mean.

$$P_{\text{macro}} = \frac{1}{|K|} \sum_{k \in K} P_k \quad R_{\text{macro}} = \frac{1}{|K|} \sum_{k \in K} R_k$$

- Macro-averaging is sensitive to outlier classes!
(can enforce balance, but also cause problems)
- **Macro-averaged Recall** is least sensitive to imbalance.

IT UNIVERSITY OF COPENHAGEN



Data Augmentation and Self-Training

IT UNIVERSITY OF COPENHAGEN

Data augmentation and self-training

- ▶ Add more data to the training set to make the classifier perform well more broadly.
- ▶ Enforce desired model symmetries to reduce overfitting
 - ▶ In computer vision: Translational or rotational invariance
 - ▶ In NLP: Upper-/lowercasing, punctuation normalisation
 - ▶ but can be more easily addressed in preprocessing
- ▶ More relevant: **Increase coverage of vocabulary and contexts**

IT UNIVERSITY OF COPENHAGEN

N-gram models for data selection

- ▶ N-gram models can measure closeness to a “model” text type.
- ▶ We can find examples that are similar to the training corpus by extracting the items with the lowest perplexity under a model trained on your reference corpus.
- ▶ How to get annotations?
 - ▶ Manual annotation: Reliable but time-consuming.
 - ▶ Self-training: Use classifier to get annotations.



IT UNIVERSITY OF COPENHAGEN

Exercise: Data Augmentation

- ▶ Find a large corpus that might contain some suitable examples for data augmentation in your task.
 - ▶ One good source: <https://opus.nlpl.eu/>
 - ▶ You will be filtering the corpus, so it's enough if there are *some* good examples in the data.
- ▶ Train an n-gram model on your TweetEval training corpus.
- ▶ Compute the perplexities of sentences in your augmentation corpus.
- ▶ Plot the distribution of sentence perplexities and inspect the examples you get with different perplexity cutoffs.
- ▶ Try self-training once you have a classifier.

IT UNIVERSITY OF COPENHAGEN

Features

IT UNIVERSITY OF COPENHAGEN

“High-level” vs. “low-level” features

- ▶ “High-level” or “engineered” features start from a linguistic hypothesis.
- ▶ Task-dependent: Good hypotheses may be hard to come by.
- ▶ “Low-level” features:
Features directly derived from the raw text, without expert knowledge of the task.
- ▶ We hope the system will learn by itself, but you'd better still have a hypothesis about why these features could be useful.

IT UNIVERSITY OF COPENHAGEN

High-level features: Coreference resolution

- ▶ Distance between expressions (measured in sentences)
- ▶ Is the first mention a pronoun? (yes/no)
- ▶ Is the second mention a pronoun? (yes/no)
- ▶ Do the two string match, after removing determiners? (yes/no)
- ▶ Does the second mention start with the determiner *the*? (yes/no)
- ▶ Does the second mention start with one of the demonstrative pronouns *this, these, that, those*?
- ▶ Do the noun phrases agree in number (singular/plural)? (yes/no)
- ▶ Do both noun phrases have the same *semantic class* (female, male, organisation, location, date, time, etc. – determined separately)? (yes/no)
- ▶ Do the noun phrases agree in gender? (yes/no/unknown – determined with word lists and prefixes such as *Mr./Mrs.*)
- ▶ Are both noun phrases proper names? (yes/no)
- ▶ Are the noun phrases known aliases of each other? (yes/no – determined with rules and lexica)
- ▶ Are the noun phrases in apposition? (yes/no – determined with rules)

(Soon et al., *Computational Linguistics* 2001)

Can you think of any high-level features that might be useful for your task?

IT UNIVERSITY OF COPENHAGEN

Bag of words

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

(Wikipedia)

IT UNIVERSITY OF COPENHAGEN

Bag of words

(((())) , , . . IR In The a a a a and and as as bag bag-of-words but disregarding document even grammar in information is its keeping language model model multiplicity multiset natural of or order processing representation represented retrieval sentence simplifying such text the this used word words

(Wikipedia)

IT UNIVERSITY OF COPENHAGEN

More low-level features

- ▶ Classes of words
 - ▶ manually curated (e.g., WordNet)
 - ▶ automatically generated (e.g., Brown clustering)
<https://pypi.org/project/brown-clustering/>
- ▶ Linguistic analysis
 - ▶ Parts of speech, syntactic relations, etc.
 - ▶ Words with dependency heads (or similar)
 - ▶ ...

IT UNIVERSITY OF COPENHAGEN

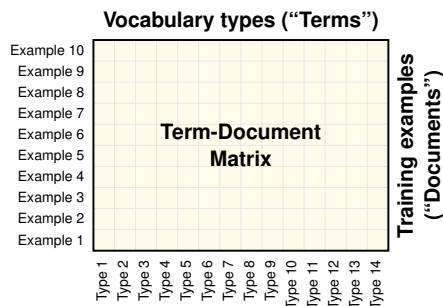
Pattern from large corpora

- ▶ Define a list of patterns to find sentences (or fragments) relevant to your task. E.g., SADNESS:
 - ▶ X was sad about PHRASE
 - ▶ X complained about feeling WORD
 - ▶ ...
- ▶ Go through **large** corpus to find matching examples.
 - ▶ Find corpora here: <https://opus.nlpl.eu/>
 - ▶ Use something like this:

```
with open('corpus.txt', 'r') as f:
    for line in f:
        if re.match(your_pattern, line):
            # find relevant part of sentence, extract
```
- ▶ Match words (or n-grams) in your dataset against words in the harvested examples.
 - ▶ Possibly exclude function words etc.
 - ▶ As features, use number of words (or n-grams, etc.) matching the corpus examples.
 - ▶ Alternatively, create artificial training examples and add them to your training set.

Term-Document Matrix

Basic feature representation:
Counts of features per training example



```
sklearn.feature_extraction.text.CountVectorizer
```

IT UNIVERSITY OF COPENHAGEN

Feature transformations

- ▶ Binary indicator features:
Presence or absence of word (encoded as $\{0, 1\}$ or $\{-1, +1\}$)
- ▶ Log-transformed counts: $\log(1 + f)$
- ▶ tf-idf transform (term frequency–inverse document frequency)
 - ▶ higher weight for frequent terms (tf)
 - ▶ lower weight for terms that occur in many documents (idf)
(will penalise function words)
 - ▶ `sklearn.feature_extraction.text.TfidfTransformer`

IT UNIVERSITY OF COPENHAGEN

Discrete vs. continuous representations

- ▶ Typical word representation is *discrete*.
 - ▶ Distinct words are considered *fundamentally different* and *incomparable*.
 - ▶ Represented as separate dimensions in a vector space.
- ▶ But words do have meaningful relations.
 - ▶ Semantic relations:
 - ▶ Hyponymy (animal – dog, object – piece of furniture – table)
 - ▶ Meronymy (part/whole: door – handle, car – wheel)
 - ▶ Antonymy (love – hate, open – close)
 - ▶ Morphological relations:
 - ▶ Inflection: take – took – taken, door – doors
 - ▶ Derivation: improve – improvement, surprise – unsurprisingly

IT UNIVERSITY OF COPENHAGEN

Word embeddings

- ▶ *Word embeddings* represent words as points in a vector space (high-dimensional, but lower dimension than vocabulary size)
- ▶ We can now measure distances between words.
- ▶ *Example:*
Principal Component Analysis of term-document matrices
 - ▶ known as *Latent Semantic Analysis* (LSA)
 - ▶ `sklearn.decomposition.TruncatedSVD`
- ▶ Frequently, we use *pretrained* word embeddings to inject *unsupervised* knowledge from large corpora.

<http://projector.tensorflow.org/>

IT UNIVERSITY OF COPENHAGEN

Word embeddings

Non-contextual word embeddings

- ▶ LSA, word2vec, GloVe.
- ▶ Represent a word as a vocabulary type.
- ▶ Word embedding is independent of context (lookup in list).
- ▶ In Python: `gensim` library.

Contextual word embeddings

- ▶ ELMo, BERT, GPT2, etc.
- ▶ Word embedding is calculated *based on specific context*.
- ▶ Neural language models.
- ▶ Very powerful (but more difficult to use).
- ▶ In Python: Huggingface `transformers` library.

IT UNIVERSITY OF COPENHAGEN