Text Analysis and Retrieval

4. Machine Learning for NLP

Prof. Jan Šnajder

University of Zagreb
Faculty of Electrical Engineering and Computing (FER)

Academic Year 2022/2023

Outline

Outline

Why machine learning?

- For many NLP tasks, it is quite difficult to come up with an algorithm that solves the task efficiently

Example NLP tasks

- POS tagging – rules that inspect the context of each word in a sentence and tag each word based on that?
- Sentiment analysis – rules that check for presence of certain sentiment-indicating words in a review?
- Named entity recognition (NER) – a manually defined finite state automaton (or an extension thereof) that recognizes sequence of words that constitute names of entities in the text?
- Semantic textual similarity – measure the word overlap and manually determine the thresholds (rules) according to which two texts are considered similar?

ML approach

- Manually label the data (supervised ML) or parts of it (semi-superivised) with labels that show how the solution looks like

Example NLP tasks

- POS tagging – manually label each word in text with its POS tag
- Sentiment analysis – manually label each review with a rating (e.g., on a scale 1–5)
- NER – manually label the spans and categories of named entity mentions in the text
- Semantic textual similarity – manually label how similar two documents are (e.g., on a scale 1–5)

- Once data is labeled, we can train a machine learning model on it
- This model can then be applied to previously unseen data to solve the NLP task in a way a human would (or almost as good)

ML approaches/tasks

- **Supervised ML:** labeled data is available for training
  - **Classification:** output is a discrete label (but there is no ordering between the labels)
  - **Regression:** output is a real-valued or integer number (obviously there is an ordering)
- **Unsupervised ML:** no labeled data is available for training
  - **Clustering**
  - **Dimensionality reduction**

Classification problems

- **Binary classification:** just two output labels (yes/no, 0/1)

$$h : \mathcal{X} \to \{0, 1\}$$

- **Multiclass classification:** each instance has one of K labels

$$h : \mathcal{X} \to \mathcal{Y}, \quad \mathcal{Y} = \{1, \ldots, K\}, \quad K > 2$$

- **Multilabel classification:** an instance can have many labels at once

$$h : \mathcal{X} \to \wp(\mathcal{Y})$$

- **Sequence labeling:** input is a sequence of instances and the output is a sequence of labels

$$h : \mathcal{X}^m \to \mathcal{Y}^m$$

- **Structured prediction:** mapping instances to structures ($\mathcal{Y}$ is a set of structures, typically $|\mathcal{Y}|$ is exponential in $|\mathcal{X}|$)

$$h : \mathcal{X} \to \mathcal{Y}$$

NLP tasks as ML problems

Sentiment analysis

Given a document, determine the overall opinion of the author.

- **problem**: binary/multiclass classification OR regression
- **input**: document (e.g., user comment, post, tweet, etc.)
- **output**: sentiment label
- **features**: e.g., bag-of-words/n-grams, emoticon/exclamation mark counts, presence of expressive lengthening ...

Named Entity Recognition (NER)

Given a document, identify and classify all named entities in text.

- **problem**: sequence labeling
- **input**: sequence of words from the sentence
- **output**: sequence of labels marking the beginnings and endings of named entities (BIO) + their category
- **features**: same as for POS, chunks, gazetteers, ...

Feature design vs. representation learning

- The key question: **how to come up with good (useful) features?**
- Two approaches:
  - **Manual feature design** – features designed based on insight or linguistic/domain expertise
    - More often: throw in everything you can (the "kitchen sink" approach), and them maybe prune later
  - **Representation learning** – features learned implicitly from data (deep learning models)

Lexical features

Lexical features

Features that encode **the identity of words**.

- lemmas or wordforms or stems
- or parts of words: e.g., suffixes, preffixes, character n-grams
- or combinations of words: e.g., bigrams, trigrams

- How to encode categorical features as numeric vectors?
  - **one-hot encoding**
  - **real-valued vectors, so-called "word embeddings"**
    $\Rightarrow$ we'll look into that in two weeks

Encoding sequences of words?

- What if we want to encode a **text fragment/sentence/document**?
- Boils down to: **how to represent the meaning of text?**
- Which boils down to: how to represent the meaning of text if we know how to represent the meaning of its words?
- Which boils down to: **semantic composition (SC)**

$\Rightarrow$ we'll look into that in the next two weeks

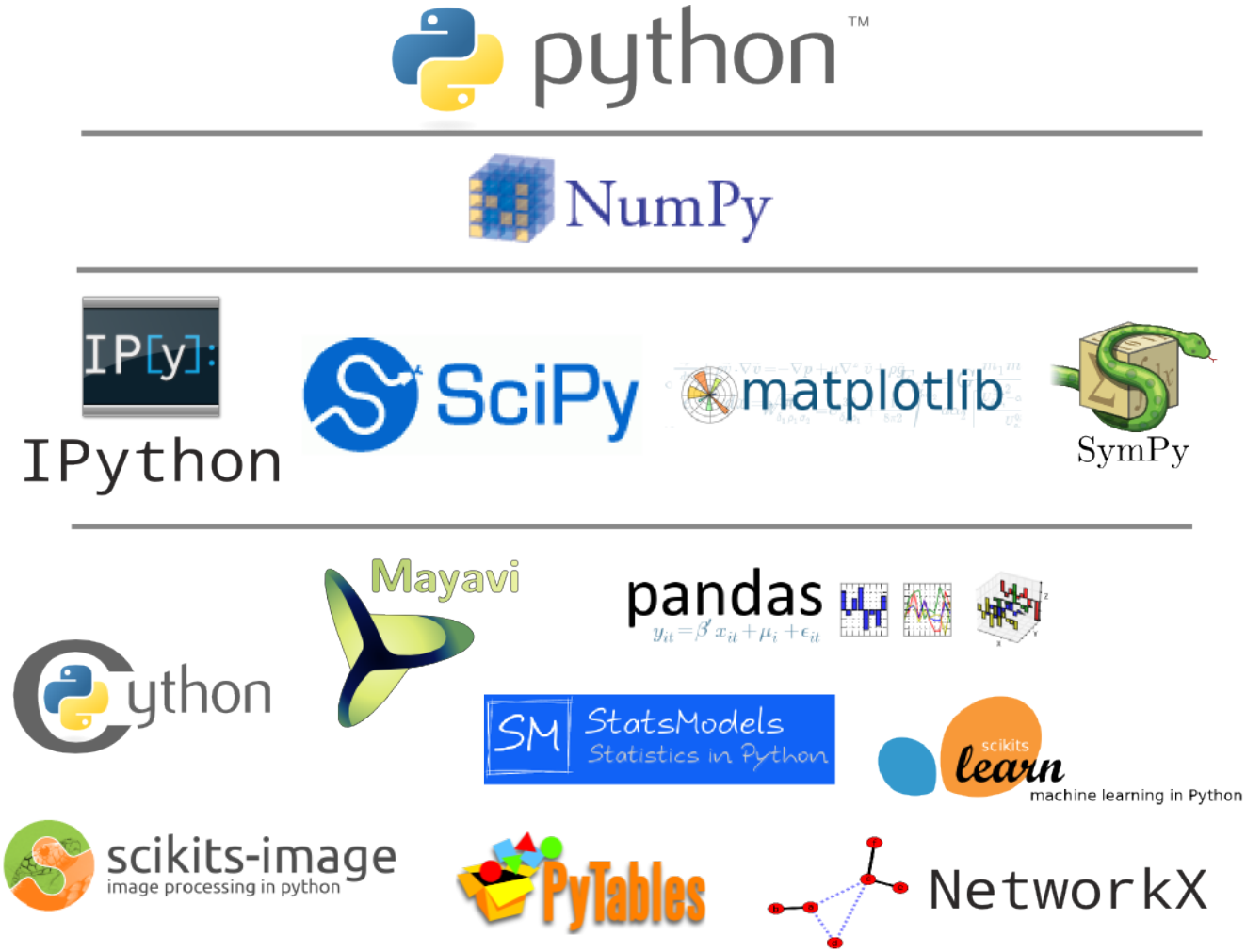
Feature analysis

- When designing features manually, will often want to see which features work and which don't.
- Why?⇒ **improved performance** and model **interpretability**
- Options:
  - **Ablation study** – turn off some features, retrain the model and see how the performance changes
  - **Feature selection** – use a method to select the best features. This can also improve the performance (especially in a "kitchen sink" approach)
- **NB**: In deep learning (aka representation learning), there are typically no features to ablate/select, but one can ablate different model components (e.g., layers, dropout, regularization, etc.)

Classifier evaluation

- To measure how well a classifier will work on unseen data, we have to evaluate it on the test set
- Standard evaluation measures (same as in IR): Accuracy, P, R, F-score
- The classifier is often compared against a baseline (a simple method that can easily be implemented). Typical baselines:
  - Majority class classifier (MCC)
  - Random classifier
  - A very simple rule-based classifier
  - A very stripped-down version of the real classifier
- To prove that one classifier is better than the other or the baseline, we have to perform statistical significance tests (typically: McNemar's test or t-test)

ML framework: scikit-learn and the SciPy ecosystem



Learning outcomes 1

1. List at least three advantages and disadvantages of machine-learning-based NLP systems
2. Explain how to frame standard NLP tasks as ML problems and what features to use
3. Explain what lexical features are and how to encode them using one-hot encoding
4. Describe the approaches to feature design and analysis

Outline

Sequence labeling

- Standard classification algorithms assume that the data points are independent ("iid")
- Many NLP problems do not satisfy this assumption: text is a sequence of words, so each word depends on the words surrounding it

Sequence labeling problem

Assigning a label (a class) to each item in a sequence. Formally:

$$h : \mathcal{X}^m \rightarrow \mathcal{Y}^m$$

Generally, the label of each token is dependent on the labels of other items in the sequence. The "iid" does not hold.
$\Rightarrow$ we need more sophisticated learning and inference techniques than for standard ML classification problems

Sequence labeling problems in NLP

Part-of-speech tagging

Mark/NNP saw/VBD the/DT saw/NN near/IN the/DT tree/NN and/CC took/VBD it/PRP to/TO the/DT tableNN.

Chunking (shallow parsing)

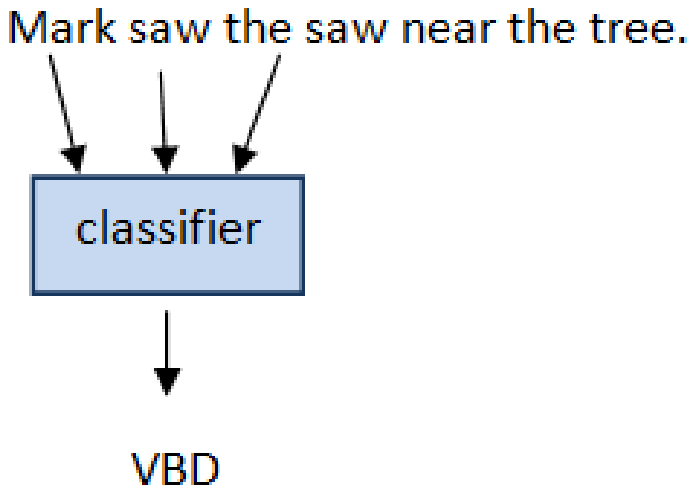
[NP Mark] [VP saw] [NP the saw] [PP near] [NP the tree] and [VP took] [NP it] [PP to] [NP the table].

Named entity recognition

Barcelona's/B-Org draw/O with/O Atletico/B-Org Madrid/I-Org at/O Camp/B-Loc NouI-Loc was/O not/O expected/O, says/O British/B-Org Broadcast/I-Org Channel's/I-Org football expert Andy/B-Per West/I-Per.

Sequence labeling as classification?

- Predict the label for each token independently, but using information from the surrounding tokens as features ("sliding window")



- Use any of the standard classification algorithm (NB, LR, SVM)
- Will this work?

Sequence labeling models

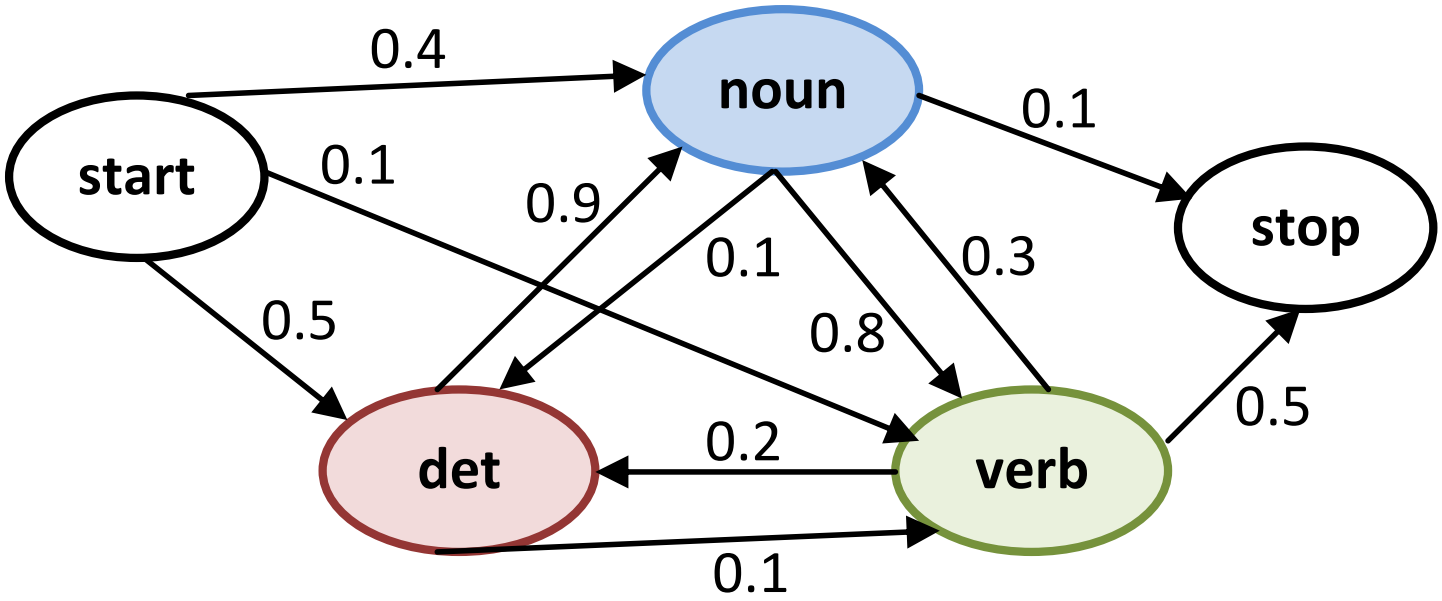
- Integrate uncertainty over multiple, interdependent classifications
- Collectively determine the most likely global assignment of labels
- Traditionally used models:
  - **Hidden Markov Model (HMM)**
  - **Maximum Entropy Markov Model (MEMM)**
  - **Conditional Random Fields (CRF)**
- More recent models:
  - **Recurrent Neural Networks (RNNs)**
    $\Rightarrow$ we'll cover these in three weeks

Hidden Markov Model

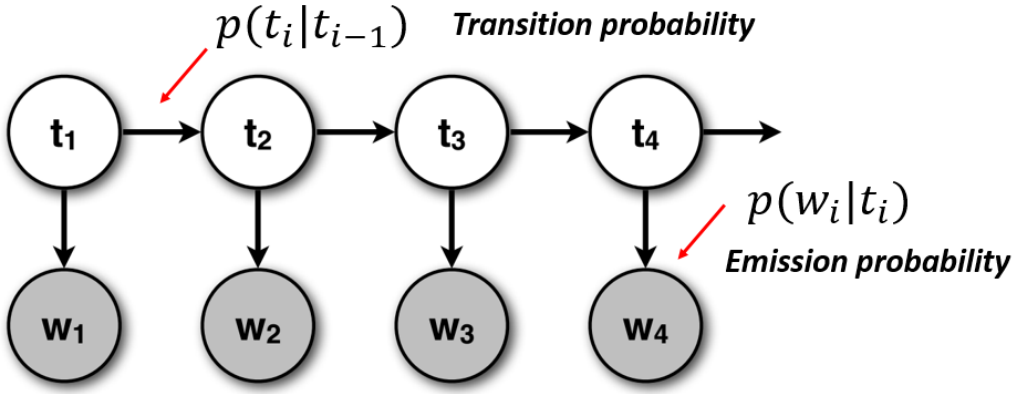
- Markov chain: the next state only depends on the current state and is independent of previous history

$$P(x_n|x_1^{n-1}) = P(x_n|x_{n-1})$$

- May be represented as a finite state machine with probabilistic state transitions



Hidden Markov Model (HMM)



$$P(t_1, \ldots, t_n | w_1, \ldots, w_n) = P(\mathbf{t}|\mathbf{w}) \propto \prod_{i=1}^{n} P(t_i|t_{i-1}) P(w_i|t_i)$$

(In practice, special "start" and "end" symbols are introduced for t_0 and t_{n+1}, respectively, and the sequence length becomes $n + 2$)

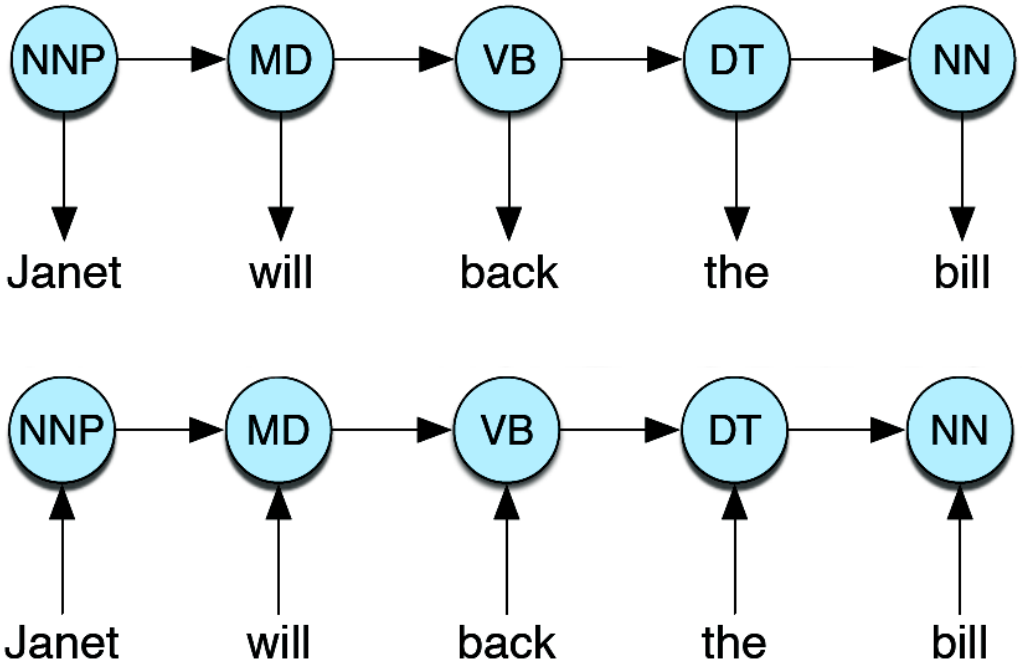
HMM with features?

- **HMM cannot encode features**, only word identities
  - adding features is possible, but the problem is that features cannot overlap due to conditional independence assumption (features themselves have to be probabilistic variables)
- **Not having features is a big drawback!**
  - e.g, in POS tagging, capitalization, suffixes, etc. tell us a lot about word's POS
- Solution: move from a generative model (HMM) to a discriminative model) model that models $P(\mathbf{t}|\mathbf{f}(\mathbf{w}))$
  - features $\mathbf{f}(\mathbf{w})$ can be arbitrary features and they may overlap

$\Rightarrow$ **Maximum entropy Markov model (MEMM)**

HMM vs. MEMM

$$P(\mathbf{t}|\mathbf{w}) \propto \prod_{i=1}^{n} P(t_i|t_{i-1})P(w_i|t_i)$$



$$P(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{n} P(t_i|t_{i-1}, w_t)$$

Conditional Random Field (CRF)

- MEMM suffers from the **"label bias problem"**
- CRF does a **global normalization** instead of local normalization:

$$\mathbf{t}^* = \arg\max_{\mathbf{t}} P(\mathbf{t} \mid \mathbf{w})$$

- Model:

$$P(\mathbf{t} \mid \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp\left\{ \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(t_{i-1}, t_i, \mathbf{w}, i) \right\}$$

- $\boldsymbol{\lambda}$ is the a vector of feature weights
- f_j is the feature function for feature j
- $Z(\mathbf{w})$ is the partition function
  $Z(\mathbf{w}) = \sum_{\mathbf{t}} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(t_{i-1}, t_i, \mathbf{w}, i)$

Learning outcomes 2

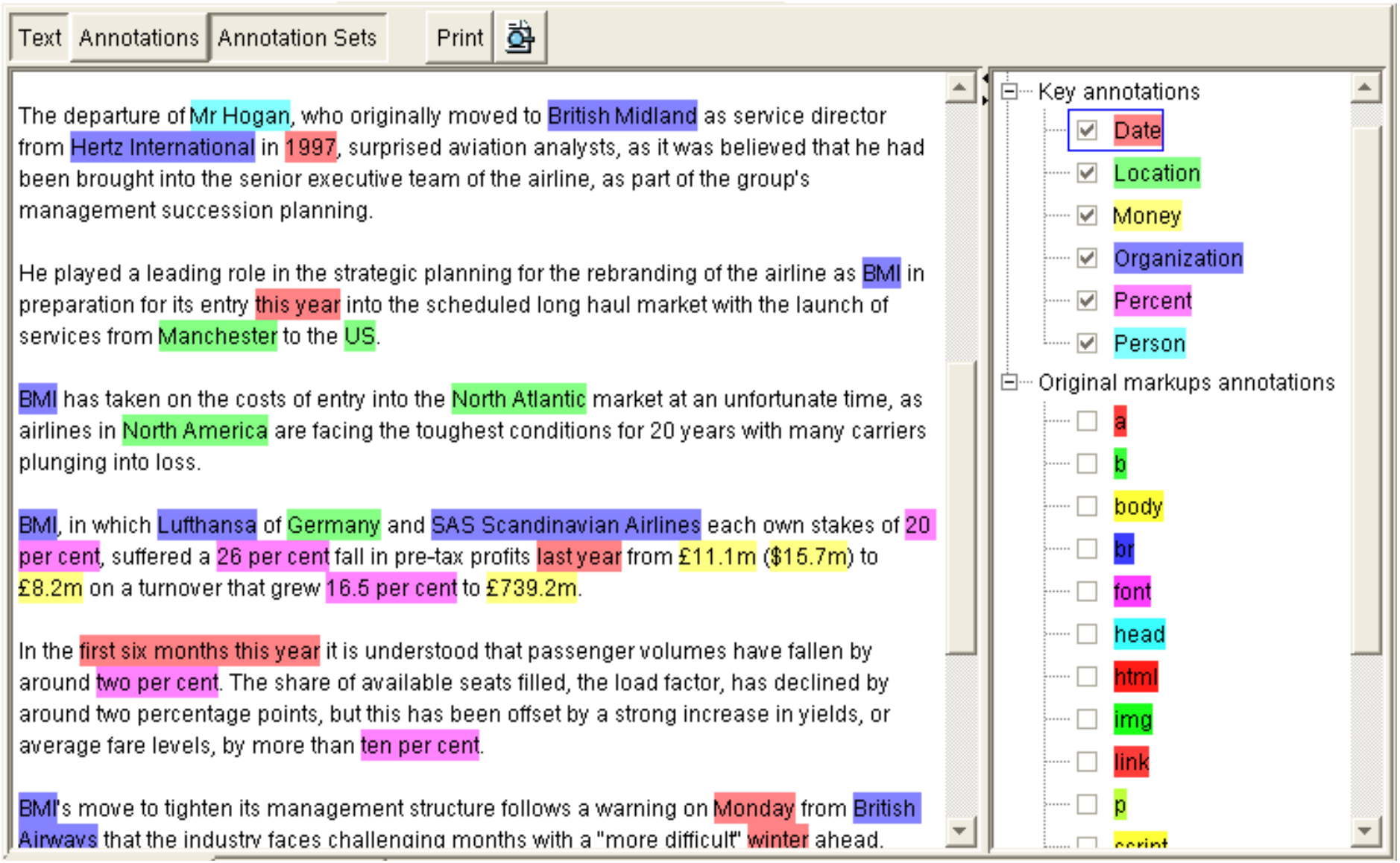
1. Explain what sequence labeling is and why we need it
2. Explain the basic idea behind HMM and its main weakness
3. Explain the basic idea behind MEMM and how it differs from HMM
4. Explain the basic idea behind CRF and how it differs from MEMM

Outline

Dataset annotation

- We often need to manually label the data for model training and evaluation. In NLP, this is called **data annotation**
  - E.g., label the POS of a word, rate how similar two words are, construct a parse tree
- The crucial question: are the annotations **correct**?
- Often the task is subjective and there is no "ground truth"
- Instead of measuring correctness, we measure **annotation reliability**: do humans consistently make the same decisions?
  - Assumption: high reliability implies validity
- Reliability is measured via **inter-annotator agreement (IAA)**

Annotation tool



Agreement

Data instance	A1	A2
wire – job	No	No
needle – locomotive	**Yes**	**No**
cake – switch	No	No
book – sky	No	No
sky – cloud	Yes	Yes
scissor – stone	**No**	**Yes**
fish – politician	No	No
thought – water	No	No
tree – war	No	No
mayor – fountain	**No**	**Yes**

- Agreement is 70%
- **Q:** Is this good enough?

Cohen's kappa

Data instance	A1	A2
wire – job	No	No
needle – locomotive	**Yes**	**No**
cake – switch	No	No
book – sky	No	No
sky – cloud	Yes	Yes
scissor – stone	**No**	**Yes**
fish – politician	No	No
thought – water	No	No
tree – war	No	No
shield – force	**No**	**Yes**

$$\begin{array}{lcc} A1 \backslash A2 & \text{Yes} & \text{No} \\ \text{Yes} & 1 & 1 \\ \text{No} & 2 & 6 \end{array}$$

$$A_o = p_{11} + p_{22} = \frac{1}{10} + \frac{6}{10} = 0.7$$

$$\begin{aligned} A_e &= p_{1\cdot}p_{\cdot 1} + p_{2\cdot}p_{\cdot 2} \\ &= \frac{2}{10} \cdot \frac{3}{10} + \frac{8}{10} \cdot \frac{7}{10} = 0.62 \end{aligned}$$

$$\begin{aligned} \kappa &= \frac{A_o - A_e}{1 - A_e} \\ &= \frac{0.7 - 0.62}{1 - 0.62} = 0.21 \end{aligned}$$

IAA and ML

- IAA defines the **upper bound (topline)** of a ML method
  - the **baseline** defines the lower bound
- Model trained on low-quality annotations will not work well (the GIGO effect)
- If IAA is too low, you should revise/aggregate the annotations
- The revised dataset is called the **gold set**

Typical annotation workflow

Data annotation

1. Prepare the data set and split it into a **calibration set** and a **production set** (several portions, perhaps overlapping)
2. Define **annotation guidelines**

   Calibration:

3. Annotators independently annotate the calibration set
4. Compute the IAA
5. Discuss the disagreements and revise guidelines if necessary
6. If IAA was unsatisfactory, repeat from step 3
7. **Production:** Annotators independently annotate the production set (each annotator one portion)
8. If portions overlap, compute the IAA (this is the IAA to report)
9. Obtain the **gold standard** by aggregation/resolving/consensus

Learning outcomes 3

1. Describe the typical annotation workflow
2. Compute and interpret the kappa coefficient for a given confusion matrix

Study assignment

1. Watch TAR "Machine learning for NLP" video lectures:
   - https://youtu.be/IUDnaExgJMA
   - https://youtu.be/Z4JaXG89AdA
   - https://youtu.be/uuwIrRTX6zw
2. Read the annotated printout of David Batista's blogs on HMM/MEMM/CRF; first 19 pages from:

   https://www.fer.unizg.hr/_download/repository/TAR-2020-reading-03.pdf
3. Self-check against learning outcomes!