Text Analysis and Retrieval

4. Machine Learning for NLP

Assoc. Prof. Jan Šnajder

With contributions from
dr. sc. Goran Glavaš

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

Academic Year 2019/2020

v2.3

Outline

Outline

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 use one-hot encoding
4. Describe the approaches to feature design and analysis

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?

Problems with rule-based systems

1. We just need too many rules to cover all the cases
2. There are many exceptions that need to be handled
3. We need expert knowledge (a linguist)
4. Difficult to design: rules interact in unpredictable ways
5. Difficult to maintain: adding new rules can easily break everything
6. Difficult to adopt to new domains: we need to modify/add rules
7. Sometimes we don't even know how to define the rules and we need to rely on intuition (aka heuristics). E.g., how much word overlap is needed to consider two texts as semantically similar?
8. Some tasks (especially the semantic ones) are inherently subjective: it is difficult to model subjectivity with rules (rules are strict)

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 approach – advantages

1. It is often much easier to manually label something than to figure out an algorithm that does the same automatically
2. Often labeling does not require (much) expert knowledge
   - E.g., most native speakers can POS tag a text in their language (perhaps with some minimal training), and most will be able to judge the similarity of a pair of documents
3. We don't care about how complex/subjective the task is: we let the "data speak for itself" and we let the ML algorithm do the work
4. If we are lucky, the labeled data will be readily available (e.g., review ratings)
5. There is tons of data out there. We can always label more data to get a better working system
6. Generally, labeling is cheaper than paying an expert

ML approach – disadvantages

1. Data labeling can be expensive, especially if large amounts of data are required and/or expert (linguistic) knowledge
   - E.g., sentiment analysis vs. POS tagging vs. parsing
2. Data labeling can be tedious (= slow, error-prone)
3. Sometimes, it requires quite of lot of training/coaching and many discussion rounds to settle the annotation disagreements
4. Unlike the rules, ML models are difficult to interpret (it's just a bunch of parameters)
   - Depends on the model (PGM $\Rightarrow$ very interpretable, deep learning $\Rightarrow$ mostly uninterpretable, essentially a "black-box approach")
5. System performance heavily depends on features used, and sometimes it's difficult to come up with sensible features for a task
   - Not a problem if we use an end-to-end deep learning model!
6. It's difficult to perform "small tweaks" on the system: we can't add a couple of rules to fix something to make the user happy

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

Document classification

Given a text document, assign to it one or more predefined categories.

- **problem**: binary/multiclass/multilabel classification
- **input**: document
- **output**: document category
- **features**: e.g., bag-of-words from the document and the title

Sentiment analysis

Given a document, determine the overal 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 ...

NLP tasks as ML problems

POS tagging

Given a sentence, determine the part-of-speech of each word.

- **problem**: sequence labeling
- **input**: sequence of words from the sentence
- **output**: sequence of POS tags from the tagset
- **features**: current/previous word identities, suffixes, capitalization, ...

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, ...

NLP tasks as ML problems

Keyphrase extraction

Given one or more documents, extract a set of content-describing phrases.

- **problem**: binary classification OR sequence labeling
- **input**: keyphrase candidates OR sequence of words from the sentence
- **output**: yes/no label OR a sequence of yes/no labels
- **features**: phrase tf-idf, position within the document, ...

Semantic textual similarity

Given a pair of text fragments, determine how similar in meaning they are.

- **problem**: regression
- **input**: a pair of texts to be compared
- **output**: similarity prediction on a 1–5 scale
- **features**: (weighted) word overlap, entity overlap, cosine and other similarities, text lengths, ...

NLP tasks as ML problems

Coreference resolution

Given text with names/pronouns, determine wheter they co-refer.

- **problem**: structured prediction $\Rightarrow$ pairwise binary classification
- **input**: a pair of mentions (names or pronouns)
- **output**: yes/no label
- **features**: bag-of-words/dependency path between mentions, ...

Dependency parsing

Given a sentence, construct its dependency parse tree with labeled edges.

- **problem**: structured prediction $\Rightarrow$ pairwise multiclass classification
- **input**: two words
- **output**: syntactic relation between them
- **features**: words (lemmas and wordforms), POS, parser state, ...

Features in NLP

- Types of features:
  - **Numeric** (e.g., distance between two words, document length), either real-valued or integer
  - **Binary** (e.g., is the word capitalized?)
  - **Categorical (multivalued)** = many discrete values without ordering (e.g., words, POS tags)

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

- Some models can work with categorical features directly (e.g., Naïve Bayes), while some work with numeric vectors only (e.g., logistic regression, SVM, deep learning models)
- 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

One-hot encoding

- Why not encode categorical features as numbers from $\{1, \ldots, K\}$?
- Not a good idea because values should remain unordered (equidistant)
- Categorical features should be encoded using **one-hot encoding**
- K-dimensional binary vector with only one component set to 1

$$\begin{aligned}
\text{word}_1 &\rightarrow \text{value } 1 \rightarrow (1, 0, 0, \ldots, 0, 0) \\
\text{word}_2 &\rightarrow \text{value } 2 \rightarrow (0, 1, 0, \ldots, 0, 0) \\
&\vdots \\
\text{word}_K &\rightarrow \text{value } K \rightarrow (0, 0, 0, \ldots, 0, 1)
\end{aligned}$$

- Vector dimension equals the number of values the feature can take. For words, this will often be on the order of ten thousands (the total size of the vocabulary)

Encoding sequences of words?

- What if we want to encode a **text fragment/sentence/document**?
  - coreference resolution $\Rightarrow$ text fragments between name mentions
  - semantic textual similarity $\Rightarrow$ the entire sentence
  - document classification $\Rightarrow$ the entire 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 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)

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

Feature selection

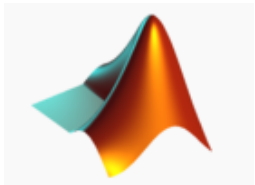
- **Filter selection**
  - Rank the features according to their relevance, and take top k features
  - Typical measures: information gain, mutual information, χ^2 statistics
- **Wrapper feature selection**
  - Iterativelly searches for a feature subset that maximizes classifier performance on the development set
- Implemented in Weka, scikit-learn, R

Classifier evaluation

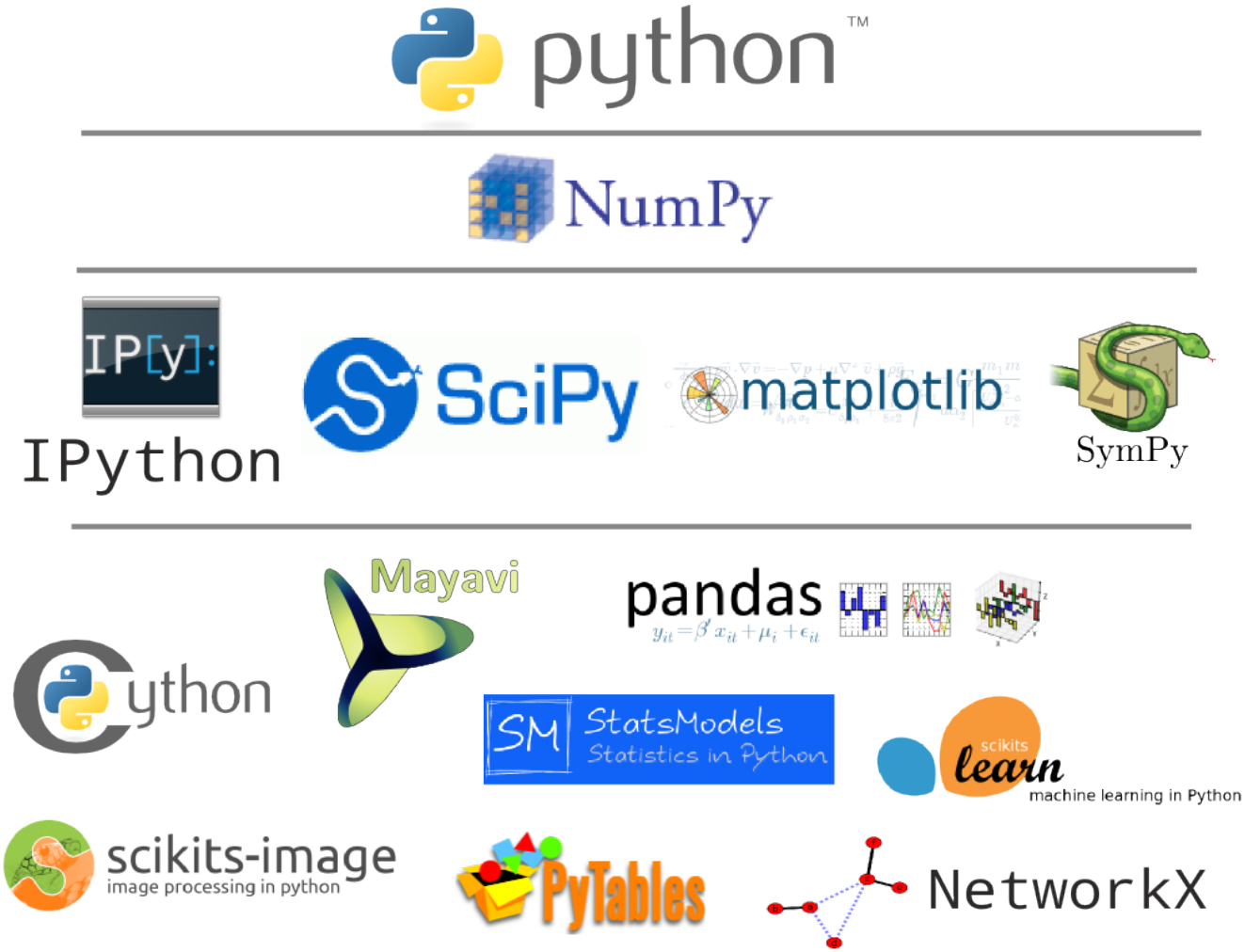
- To measure how well a classifier will work on unseen data, we have to evaluate it on the test set
- Standard evaluation measures (same as in IR): Accuracy, P, R, F-score
- The classifier is often compared against a baseline (a simple method that can easily be implemented). Typical baselines:
  - Majority 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: t-tests)

ML frameworks

- Weka (GPL)
  www.cs.waikato.ac.nz/ml/weka
- RapidMiner (AGPL)
  rapid-i.com
- Orange (GPL)
  www.ailab.si/orange
- R (GPL)
  www.r-project.org
- Matlab
  www.mathworks.com/products/matlab



ML frameworks: scikit-learn and the SciPy ecosystem



Learning outcomes 1 – CHECK!

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

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

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)

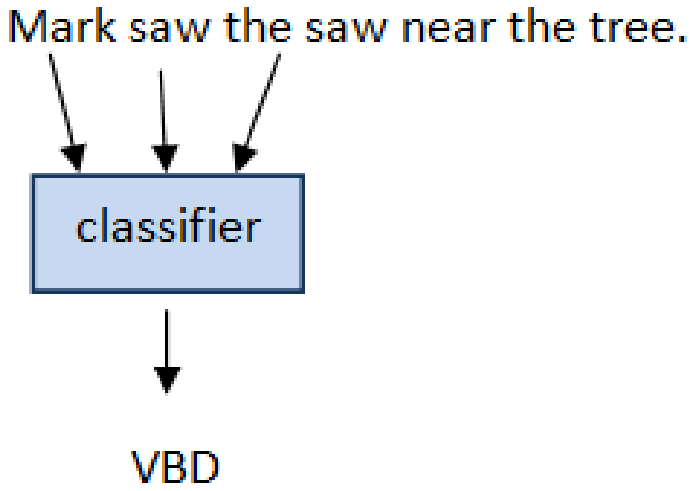
[NP Mark] [VP saw] [NP the saw] [PP near] [NP the tree] and [VP took] [NP it] [PP to] [NP the table].

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)

Sequence labeling as classification?

- Say we do POS tagging. Can we use the labels (POS tags) of the surrounding tokens as features for the classifier?
- Output labels of surrounding tokens are usually very good predictors
  - e.g., POS tags of "Mark" and "the" are indicative of the POS tag of "saw"

  $\Rightarrow$ sequence labeling as classification **with outputs as inputs**
- However, at prediction time, not all labels are available for all tokens
  - if we do left-to-right, we only have left labels (and conversely for right-to-left)
- Solution: instead of doing independent (local) decisions, determine **the most likely joint (global) assignment of labels** to all of the tokens in a sequence

Sequence labeling models

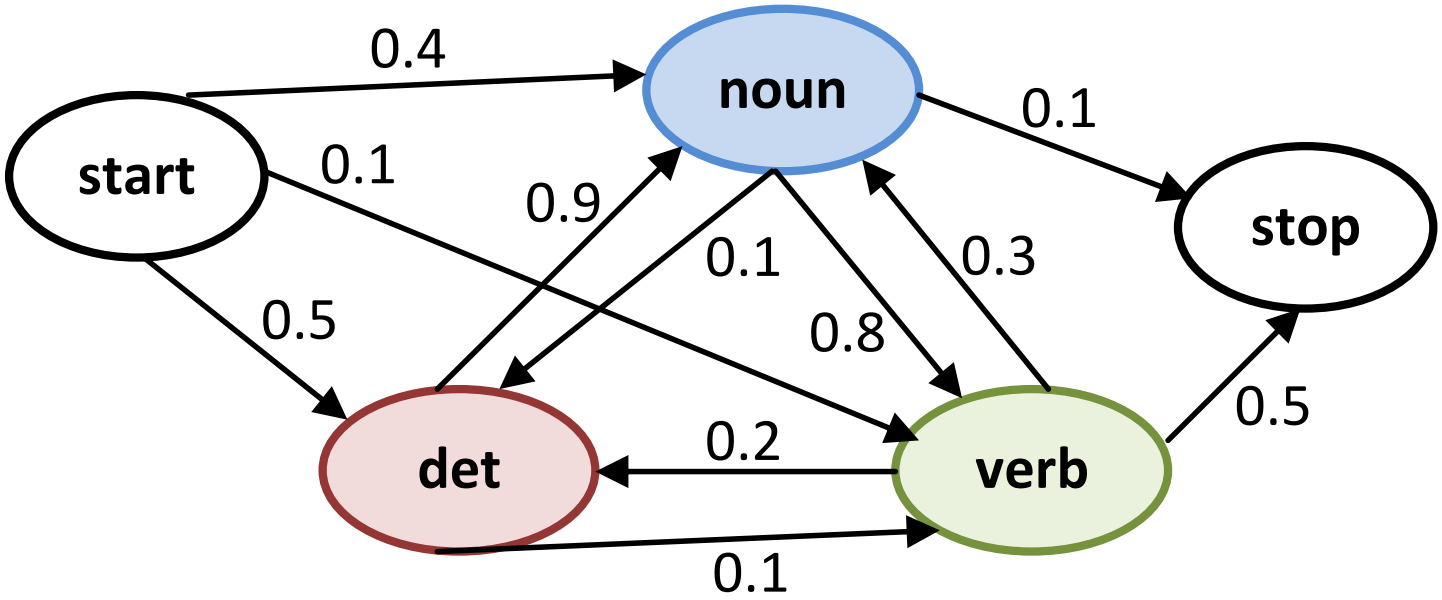
- Integrate uncertainty over multiple, interdependent classifications
- Collectively determine the most likely global assignment of labels
- Traditionally used models:
  - **Hidden Markov Model (HMM)**
  - **Maximum Entropy Markov Model (MEMM)**
  - **Conditional Random Fields (CRF)**
- More recent models: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

Hidden Markov Model

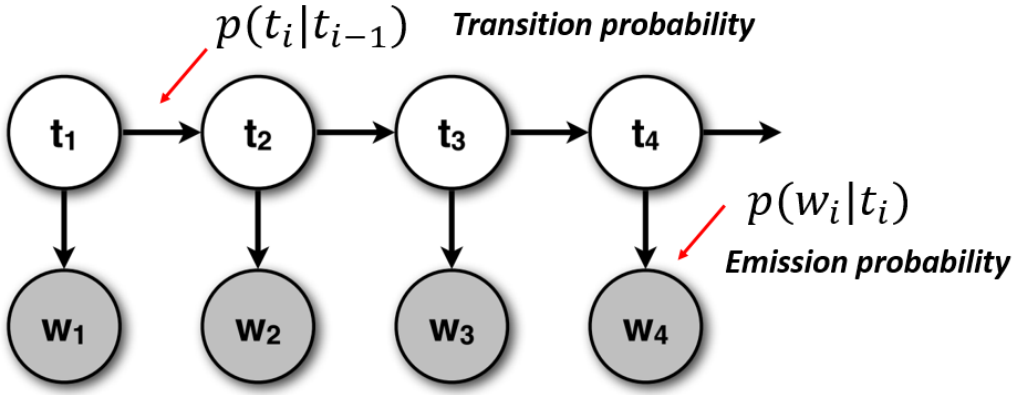
- Markov chain: the next state only depends on the current state and is independent of previous history

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

- 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 $n + 2$)

Hidden Markov Model (HMM)

- **Training**
  - Since this is a PGM, training amounts to parameter estimation
  - Parameters are the emission and transition probabilities
  - Supervised: MLE (may overfit) or MAP (e.g., add one)
  - Unsupervised: Baum-Welch algorithm (variant of EM)
  - Possible improvement: extend from bigram to trigram model (estimate $P(t_i|t_{i-1}, t_{i-2})$)
- **Predicting** (aka **decoding**)
  - Since this is a PGM, predicting amounts to probabilistic inference
  - Determine the most likely label sequence $\mathbf{t}$ for word sequence $\mathbf{w}$ $\Rightarrow$ "MAP query":

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

$\Rightarrow$ no closed-form solution, but efficiently solvable using the **Viterbi algorithm**

Viterbi algorithm

- Dynamic programming algorithm for "MAP query" in HMM
- Viterbi lattice ("trellis diagram"):

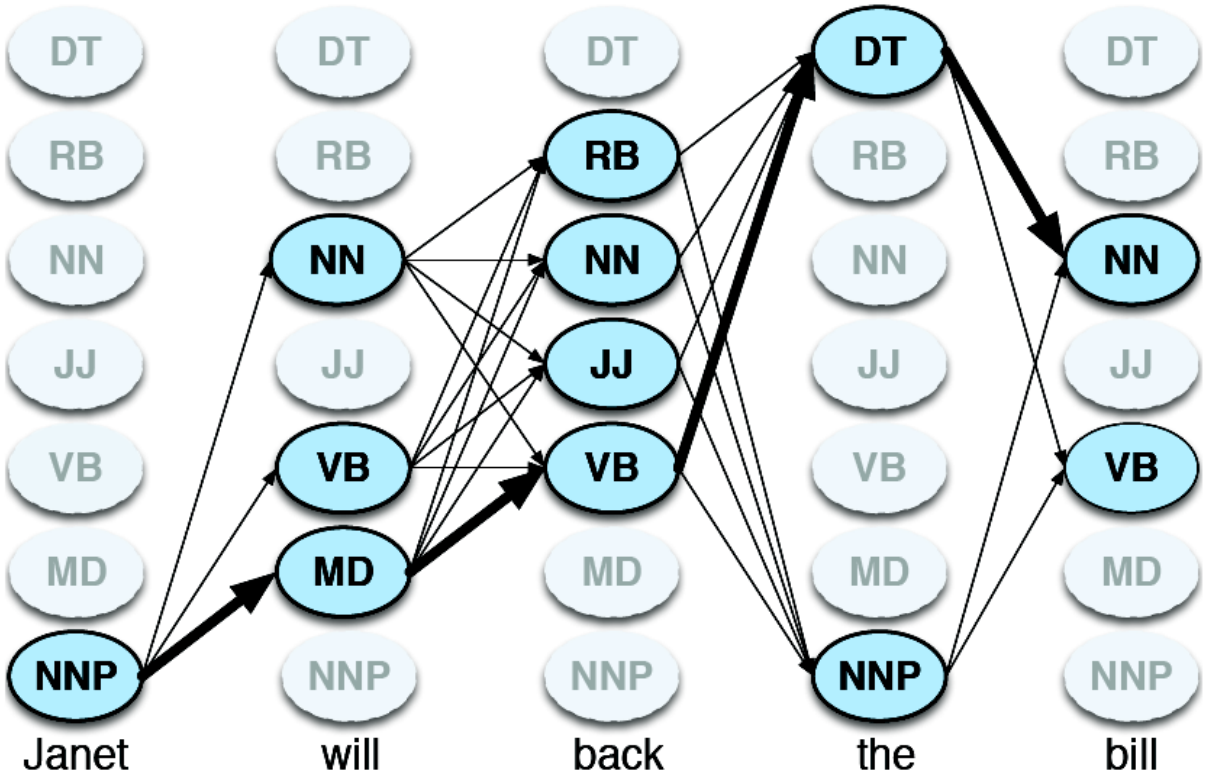


Figure from: Jurafsky & Martin, Speech and Language Processing, 2020

Beam search

- Viterbi runs in $\mathcal{O}(n|T|^2)$, where n is sequence length and $|T|$ is tagset size $\Rightarrow$ problematic for large tagsets
- The alternative is to use beam search: in each timestep, keep only the top-K ("beam width") most probable states

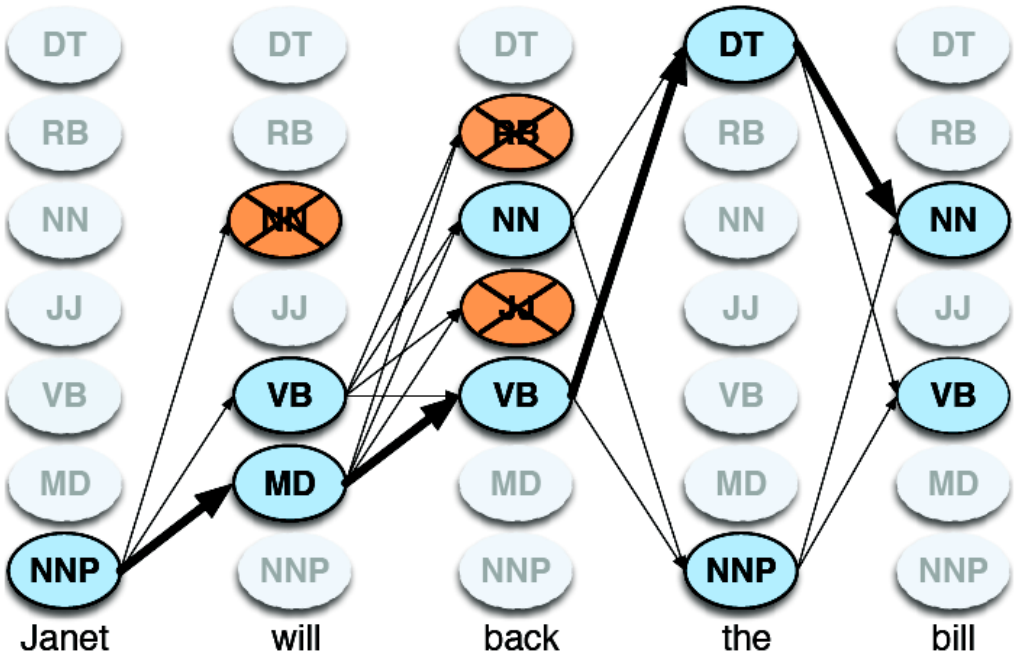


Figure from: Jurafsky & Martin, Speech and Language Processing, 2020

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 conditional (i.e, discriminative) model that models $P(\mathbf{t}|\mathbf{f}(\mathbf{w}))$
  - features $\mathbf{f}(\mathbf{w})$ can be arbitrary features and they may overlap
  - features are observed, i.e., not treated as probabilistic variables

Maximum Entropy Markov Model (MEMM)

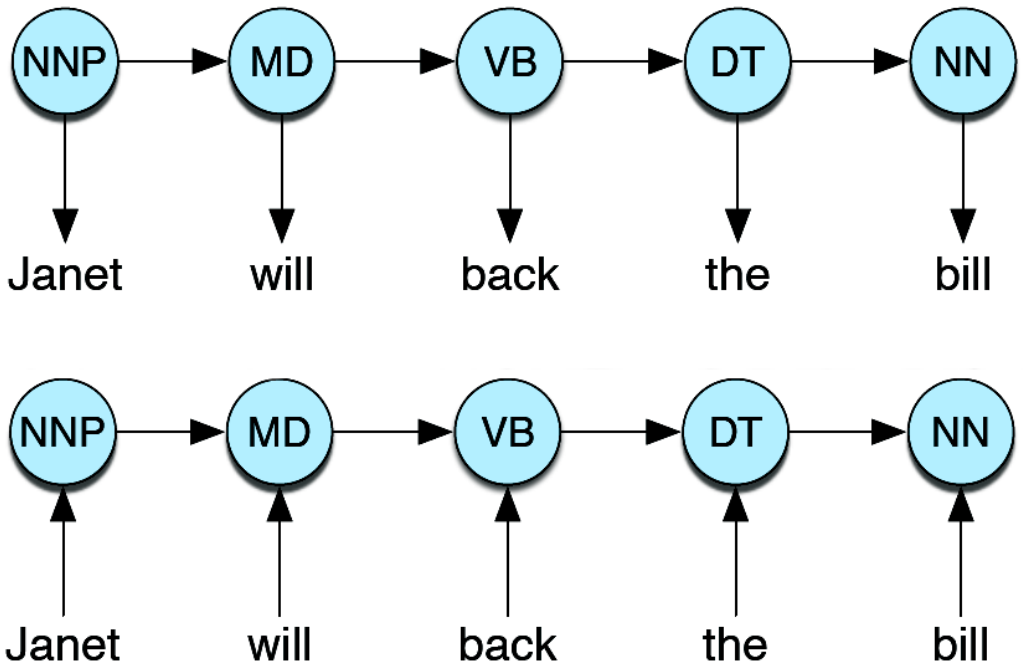
- Combines multinomial logistic regression (aka "maximum entropy") and a Markov model (conditioning on the previous state)
- Conditional modeling of label probability given previous label and the current observation:

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

- Each transition probability models the same distribution $P(t|t', w)$, modelled using multinomial logistic regression (input: word and a previous label, output: probability distribution over labels)

HMM vs. MEMM

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



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

Maximum Entropy Markov Model (MEMM)

- **Model**: decompose $P(t|t', w)$ into $|T|$ separate distributions, $P_{t'}(t|w)$, one for each input label, giving us $|T|$ multinomial logistic regression models (input: word, output: distribution over labels)
- Then, convert to a **log-linear model** and introduce **feature functions**:

$$P(t|t', w) = P_{t'}(t|w) = \frac{1}{Z(t', w)} \exp\left\{\sum_j \lambda_j f_j(t, w)\right\}$$

- λ_j are the feature weights
- f_j are feature functions (notational trick to have one weight vector for all labels)
- Z is the partition function (normalization constant to ensure probabilities sum to 1)
- **Training**: no closed-form solution $\Rightarrow$ iterative scaling or L-BFGS

Multinomial logistic regression – a refresher

- $\mathbf{x}$ and $\mathbf{w}$ are n-dimensional feature and weight vectors, respectively, and $y = \{0, \ldots, K-1\}$ are the class labels
- Model for each class:

$$h_k(\mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_l \exp(\mathbf{w}_l^T \mathbf{x})} = P(y = k|\mathbf{x})$$

  where $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_K)$ is a matrix of weights for all K classes
- The model (distribution) for all K classes:

$$P(y|\mathbf{x}) = \text{softmax}(\mathbf{W}^T \mathbf{x})$$

- We can rewrite the model for each class as:

$$P(y = k|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{j=0}^{n} w_{j,k} x_j \right\}$$

  where $Z(\mathbf{x}) = \sum_l \exp(\mathbf{w}_l^T \mathbf{x})$ is the partition function

Multinomial logistic regression – feature functions

- Let's concatenate $\mathbf{w}_1, \ldots, \mathbf{w}_K$ into one $(K \cdot n)$-dimensional vector:

$$\mathbf{w}^T = \mathbf{w}_1^T \oplus \mathbf{w}_2^T \oplus \cdots \oplus \mathbf{w}_K^T$$

- Redefine j to index from $j = 0, \ldots, K \cdot n - 1$
- Introduce feature functions, $f_j(y, \mathbf{x})$, each defined to output one feature from $\mathbf{x}$ for only one label y, and 0 otherwise:

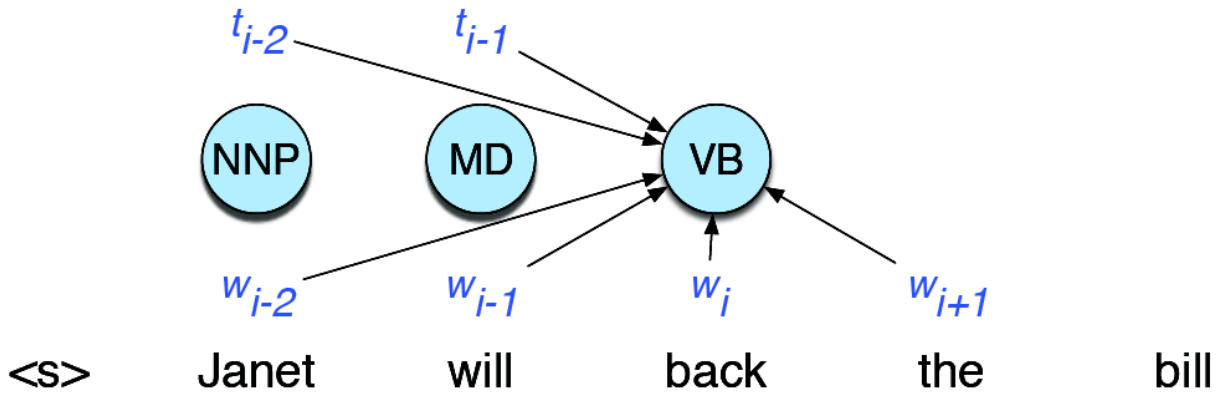
$$f_j(y, \mathbf{x}) = \mathbf{1}\{y = \lfloor j/n \rfloor\} x_{(j \bmod n)}$$

- We can now rewrite the model for all K classes as:

$$P(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{ \sum_{j=0}^{K \cdot n - 1} w_j f_j(y, \mathbf{x}) \right\}$$

Maximum Entropy Markov Model (MEMM)

- In practice, a feature function may make use of the entire input sequence $\mathbf{w}$ (all observed words, not only the current one) and all previous tags (not only one previous tag)



- How is this different from sequence labeling as classification? So far it isn't, but the difference is that the final labels are determined using decoding: a variant of the Viterbi algorithm

Label bias problem

- Each logistic regression model is **normalized locally** $\Rightarrow$ probabilities need to sum to 1
- This gives rise to the so-called label bias problem:
  1. information on the likelihood of observation gets discarded
  2. states with fewer outgoing transitions are preferred
  3. states with low-entropy transition distributions are preferred

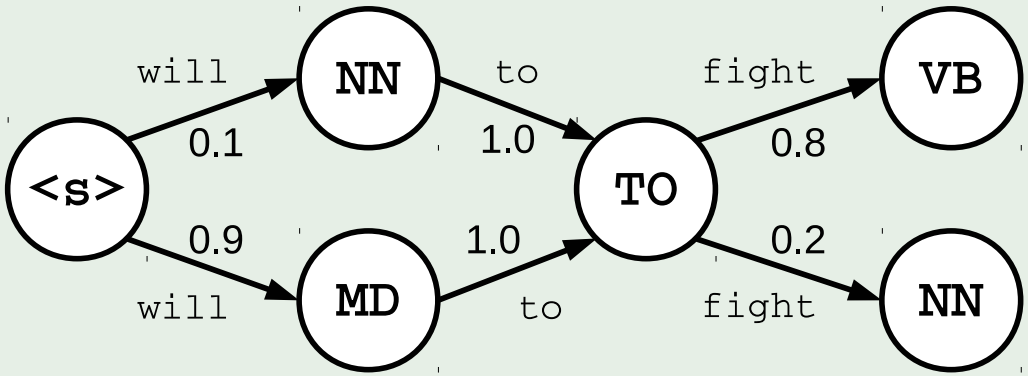
$\Rightarrow$ The max-probability path **may not correspond to globally optimal label sequence** $\Rightarrow$ poor model performance

- Read more about label bias here

Label bias problem

Example: POS tagging

Correct: `<s> will/NN to/TO fight/VB`



Incorrect: `<s> will/MD to/TO fight/VB`

Toutanova et al. (2003)

Conditional Random Field (CRF)

- A model that eliminates the label bias problem of the MEMM
- Directly models the distribution of the entire output vector (vector of labels) over the entire input feature vector $\Rightarrow$ **global normalization** instead of local normalization
- The most probable output vector is assigned as the sequence of labels for the given input vector

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

- **NB**: CRFs can be modeled as undirected PGMs (Markov Random Field = Markov Network = Undirected PGM) conditioned on $\mathbf{w}$

Conditional Random Field (CRF)

- Explicit account of the features computed on the input vector

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

- $\boldsymbol{\lambda}$ is the a vector of feature weights
- f_j is the feature function for feature j
- $Z(\mathbf{w})$ is the partition function:

$$Z(\mathbf{w}) = \sum_{\mathbf{t}} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(t_{i-1}, t_i, \mathbf{w}, i)$$

Conditional Random Fields

- Example of feature computation (POS tagging):

$$f_j(t_{i-1}, t_i, \mathbf{w}, i) = \begin{cases} 1, & \text{if } t_{i-1} = \text{'NN'} \wedge ends(w_i, \text{"ed"}) \wedge t_i = \text{'VBD'} \\ 0, & \text{otherwise.} \end{cases}$$

- Provided an input sequence $\mathbf{w}$, a variant of Viterbi can find the most likely label sequence $\mathbf{t}$ that maximizes $P(\mathbf{t}|\mathbf{w})$ under the model
- Parameters λ_j are determined in a supervised fashion, by maximizing the likelihood of manually annotated data $\mathcal{L}(\boldsymbol{\lambda}|D)$
- The optimal solution is not always computable in the closed form (depends on the features)
- Typical CRF implementations use *iterative scaling* or gradient descent-based techniques to optimize the parameters

PGMs: Generative vs. discriminative

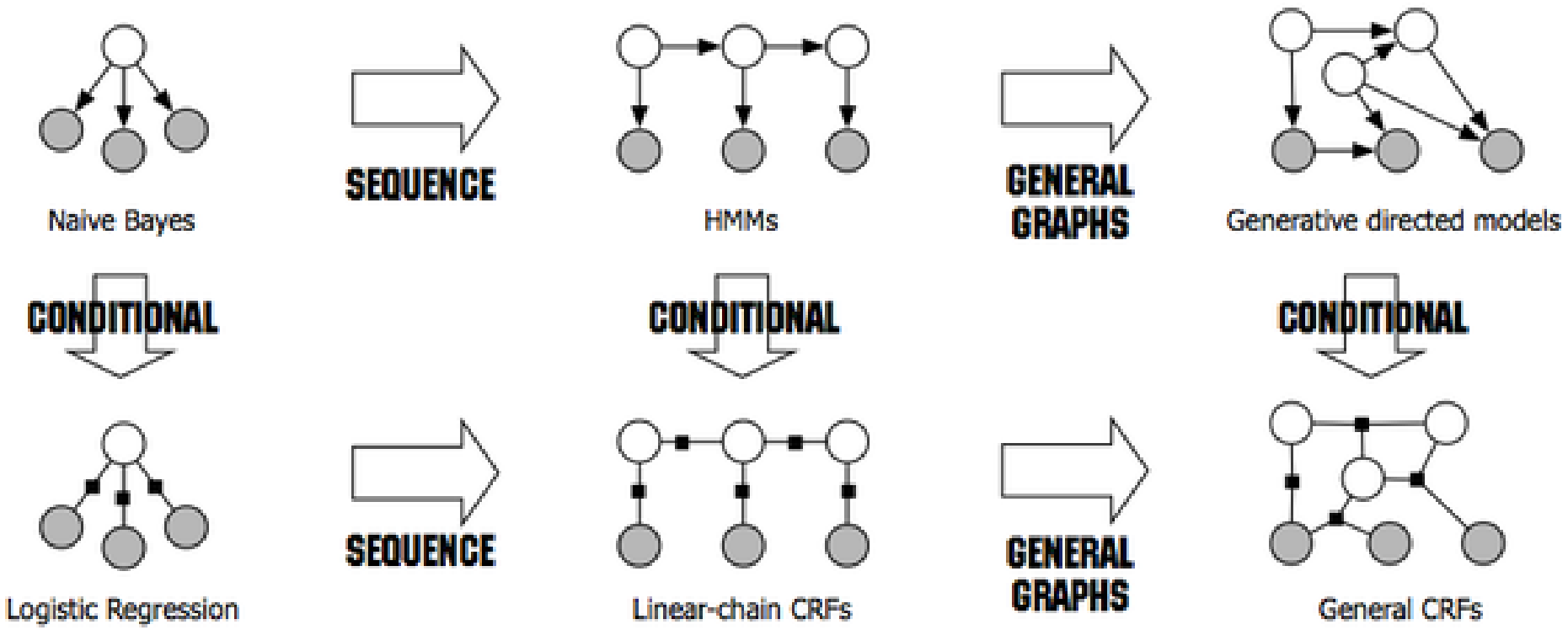


Figure 1.2 Diagram of the relationship between naive Bayes, logistic regression, HMMs, linear-chain CRFs, generative models, and general CRFs.

C. Sutton and A. McCallum. "An introduction to conditional random fields." Machine Learning 4.4 (2011): 267–373.

Learning outcomes 2 – CHECK!

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

Learning outcomes 3

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

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)**

Inter-annotator Agreement (IAA)

- Have two or more annotators annotate the same data independently but according to the same guidelines
- Measure the **inter-annotator agreement (IAA)**:
  - Agreement %
  - Cohen's Kappa
  - Fleiss' Kappa
  - P, R, F1 (results of one annotator taken as true labels)

- An excellent paper on measuring IAA in NLP:
  Artstein, R., & Poesio, M. (2008). Inter-coder agreement for computational linguistics. Computational Linguistics, 34(4), 555-596.

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?

Agreement

- Suppose annotators are notoriously uninterested and label instances at random.
  Q: How much agreement can we expect?
  - Both choose Yes: $0.5 \cdot 0.5 = 0.25$
  - Both choose No: $0.5 \cdot 0.5 = 0.25$

  $\Rightarrow$ 50% chance agreement
- Suppose both annotators label 95% of instances as No.
  Q: How much agreement can we expect?
  - Both choose Yes: $0.05 \cdot 0.05 = 0.0025$
  - Both choose No: $0.95 \cdot 0.95 = 0.9025$

  $\Rightarrow$ 90.50% chance agreement

- Obviously, IAA measures have to be corrected for chance agreement

Cohen's Kappa

- Cohen's kappa Computes the chance-corrected IAA

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

A_o – observed agreement, A_e – agreement expected by chance

- Agreements are computed from the confusion matrix

$$\begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \end{array} \xrightarrow{/N} \begin{array}{c} \\ p_{1\cdot} \\ p_{2\cdot} \end{array} \begin{array}{c} \begin{array}{cc} p_{\cdot 1} & p_{\cdot 2} \end{array} \\ \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} \end{array}$$

Observed agreement: $A_o = p_{11} + p_{22}$
Expected agreement: $A_e = p_{1\cdot}p_{\cdot 1} + p_{2\cdot}p_{\cdot 2}$

- $\kappa \in [-1, +1]$ (0 = chance agreement, 1 = perfect agreement)

Cohen's kappa – example

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**

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

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

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

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

Cohen's kappa – interpretation

- There is no agreed-upon interpretation
- Krippendorff (1980):
  - $\kappa < 0.67$ – discard
  - $\kappa \geq 0.67$ – tentative agreement
  - $\kappa \geq 0.8$ – good agreement
- Landis and Koch (1997):
  - $\kappa < 0.2$ – slight
  - $\kappa \geq 0.2$ – fair
  - $\kappa \geq 0.4$ – moderate
  - $\kappa \geq 0.6$ – substantial
  - $\kappa \geq 0.8$ – perfect

Causes of low agreement

1. Random errors (slips)
2. Misinterpretation of annotation guidelines
3. Different intuitions
4. The task is subjective
   - Objective tasks: POS tagging, parsing, NER, ...
   - Subjective tasks: semantic tasks, sentiment analysis, ...

- Ideally, we wish to eliminate the first three causes
- The remaining disagreement is due to subjectivity and indicates the intrinsic difficulty of the task

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)
  - ML model will learn to make the same mistakes, or will simply be confused due to inconsistent labels ("teacher noise")
- If IAA is too low, you should revise/aggregate the annotations
  - enforce consensus or resolve disagreements by a third party (good for objective tasks)
  - average over annotations (good for subjective tasks)
- The revised dataset is called the **gold set**

Typical workflow

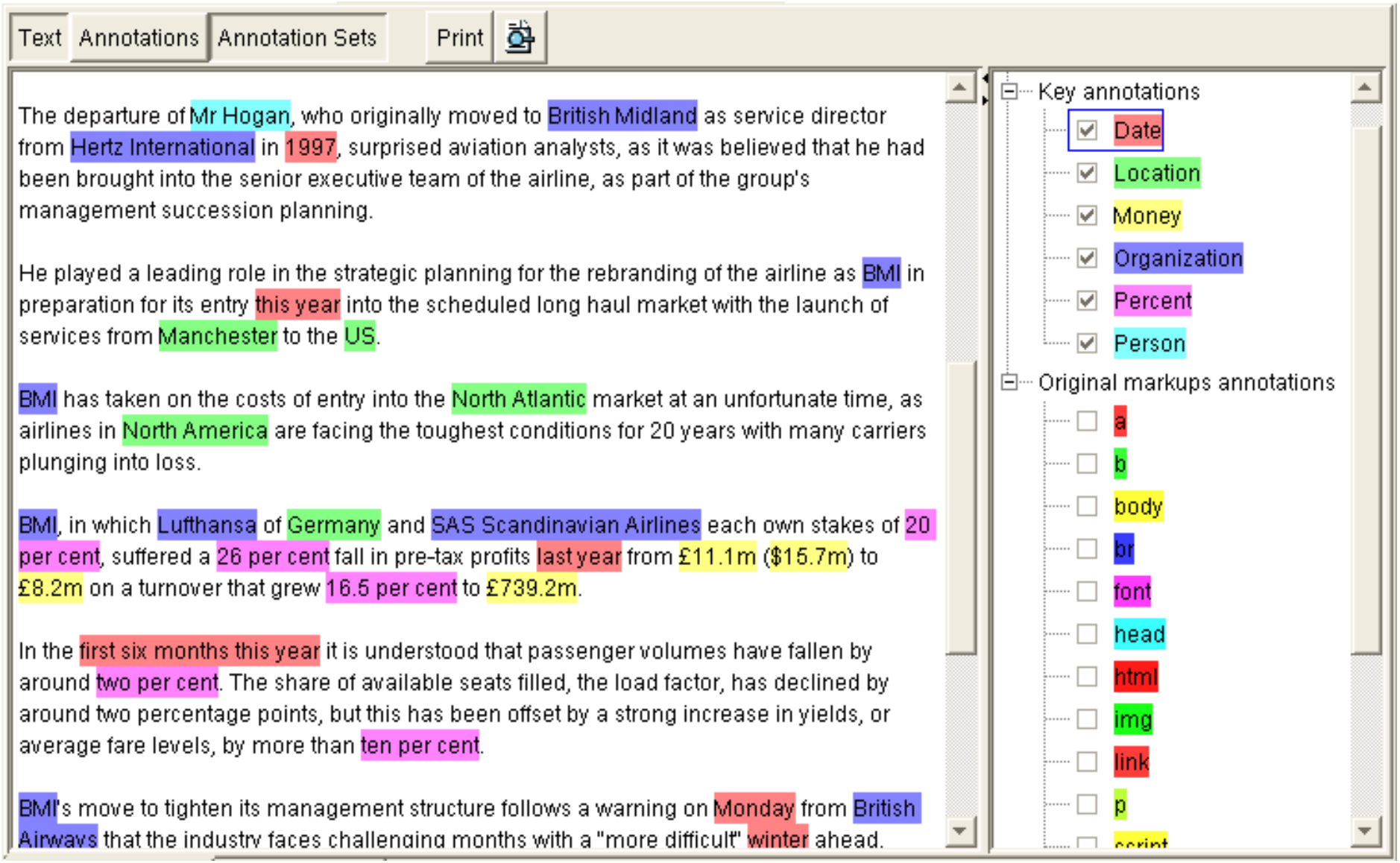
Data annotation

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

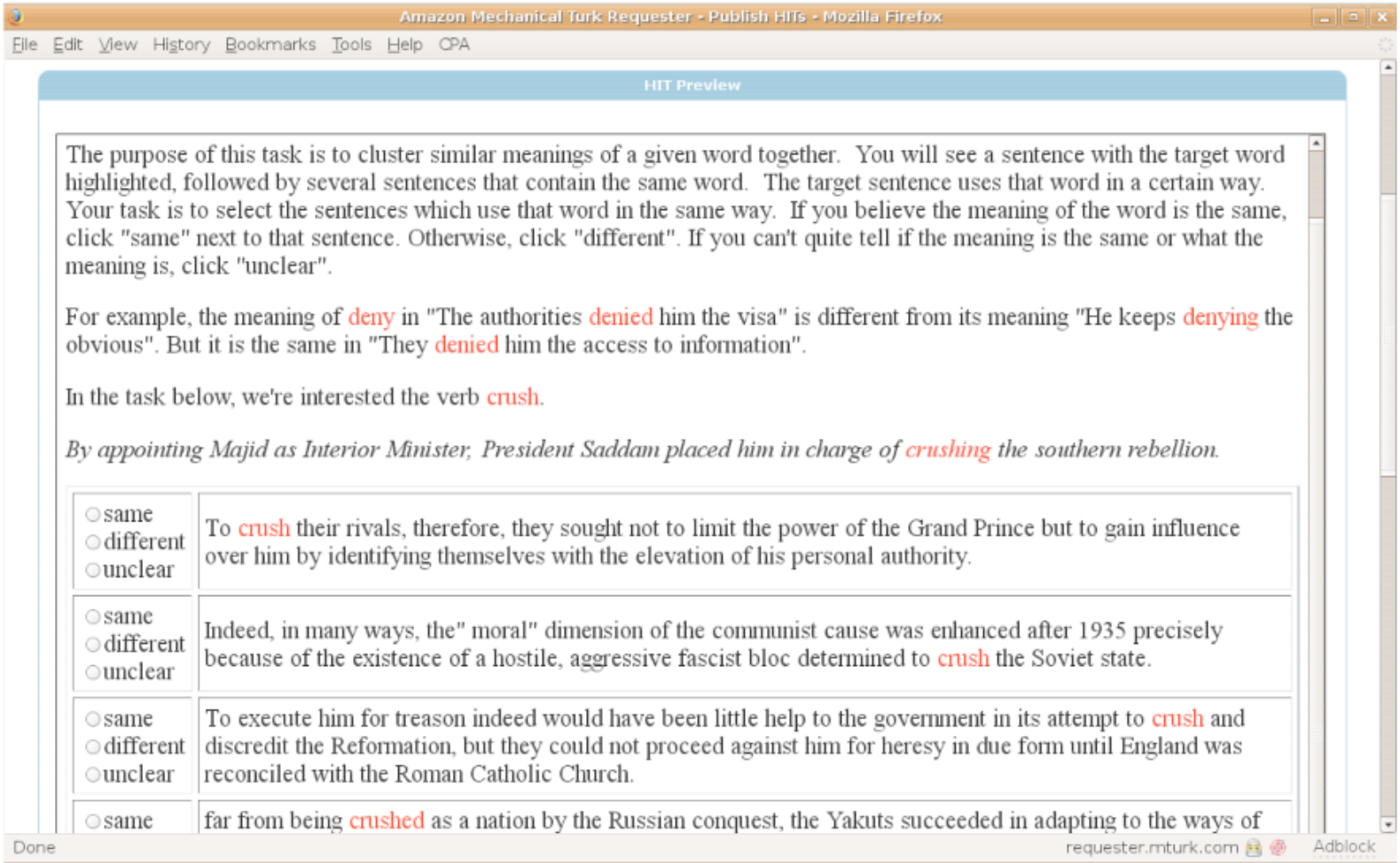
Annotation tool



Who are the annotators?

- **Students** (for course credits, fun, comradeship, or altruism)
- **Experts** (linguists, documentalists, domain experts)
- **Users** themselves (click data, feedback responses)
- **Crowdsourcing**
  - Soliciting annotations from a large group of people (online)
  - Annotators **(workers, turkers)** are payed per completed job (hit)
  - The task has to be simple!
  - CF platforms: **CrowdFlower, Amazon Mechanical Turk, ...**
    - \+ Cheap and relatively fast
    - \+ A large number of responses (good for subjective tasks)
    - – Restricted to simple tasks
    - – Generally unreliable
    - – Difficult to measure IAA and ensure quality

Crowdsourcing annotation



Rumshisky, A. (2011). Crowdsourcing word sense definition. ACL LAW-V, pp. 74–81.

Learning outcomes 3 – CHECK!

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