Statistical Natural Language Processing

Part VII: NLP using Sequence Labeling

Henning Wachsmuth

https://ai.uni-hannover.de

Learning Objectives

Concepts

- The notion of sequence labeling tasks
- BIO tagging
- Probabilistic sequence models

Methods

- Generative sequence labeling with hidden Markov models
- Discriminative sequence labeling with conditional random fields
- The Viterbi algorithm for decoding

Tasks

- Part-of-speech tagging
- Named entity recognition

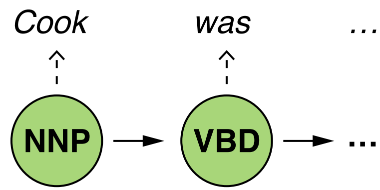
Outline of the Course

Introduction

Sequence Labeling

Sequence labeling

- The task to assign the most likely sequence of labels $(y_1, \ldots, y_n)$ to a sequence of text spans $(o_1, \ldots, o_n)$
- Each span o_i is thereby assigned the label y_i.
- The set of $k > 1$ nominal labels is predefined.



Use of sequence labeling

- Any classification of text spans in a given interrelated sequence can be modeled as a sequence labeling task.
- This is helpful when y_i may depend on text spans preceding (and/or succeeding) o_i and/or on their labels.

Sequence labeling in NLP

- Classification. Decide types of consecutive text spans.
- Identification. Find segments of text that represent specific concepts.

  Identification often also includes classification.

Sequence Labeling

BIO Tagging

Segmentation in sequence labeling

- Many tasks consist in finding segments of $k \geq 1$ tokens (or other spans) that have a specific meaning or syntactic function.

  [Facebook]$_{ORG}$ is called [Meta Platforms]$_{ORG}$ now.

- Joint identification and classification is often modeled as *BIO tagging*.

BIO tagging

- Decide for each text span whether it is at the *beginning (B)*, *inside (I)*, or *outside (O)* of a segment of a specific type.

  Variations exist, such as BIOES with additional *end (E)* and *single unit (S)* tags.

- *B* is needed for boundaries between neighboring segments.

  People/O still/O call/O Meta/B-ORG Platforms/I-ORG Facebook/B-ORG ./O

Notice

- For k class labels, BIO tagging distinguishes $2 \cdot k + 1$ tags (*O* once only).

Sequence Labeling

Selected Sequence Labeling Tasks

Part-of-speech (POS) tagging

- The task to assign a POS tag to each token in a given sentence or text

  If/IN Tim/NNP Cook/NNP was/VBD a/DT cook/NN ,/, he/PRP would/MD cook/VB ./.

Named entity recognition (NER)

- The task to identify and classify named entities of different types in a given sentence or text

  Tim/B-PER Cook/I-PER works/O in/O Cupertino/B-LOC ./O

Other tasks

- Phrase chunking. Segmenting sentences into phrases of different types
- Numeric entity recognition. Finding alphanumeric entities (e.g., dates)
- Domain and genre-specific tasks, such as mining argumentative units

Sequence Labeling

NLP using Sequence Labeling

Modeling sequence labeling

- Input. A sequence of text spans $O = (o_1, \ldots, o_k)$
- Output. A sequence of labels $Y = (y_1, \ldots, y_k)$, such that y_i refers to o_i

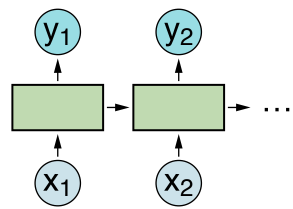
Tim Cook works in Cupertino. $\rightarrow$ (B-PER, I-PER, O, O, B-LOC, O)

Methods for sequence labeling

- Some methods consecutively label each span o_i, others label all jointly.
- y_i is predicted from o_i and preceding spans and labels $(o_{i-1}, y_{i-1}), \ldots$
- To this end, a *probabilistic sequence model* combined with a *decoding algorithm* is trained in a supervised manner.

Neural sequence labeling

- Recurrent neural networks and transformers are used for sequence labeling, too.



Details follow in Lecture Parts VIII and IX.

Sequence Labeling

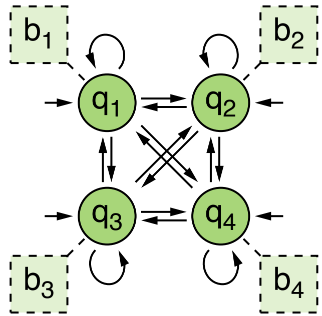
Probabilistic Sequence Models

Probabilistic sequence model

- Describes the conditional dependencies of a given set of probabilistic random variables $Q = \{q_1, \ldots, q_m\}$ as a graph
- In sequence labeling, each $q \in Q$ represents one possible label y.

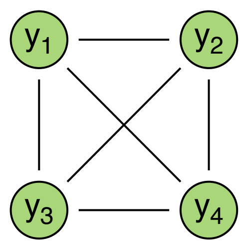
Generative models for sequence labeling

- Estimate the joint probability distribution $P(O, Q)$ of sequences of spans O and states Q
- Require much data; can also be used for generation
- Example. Hidden Markov model



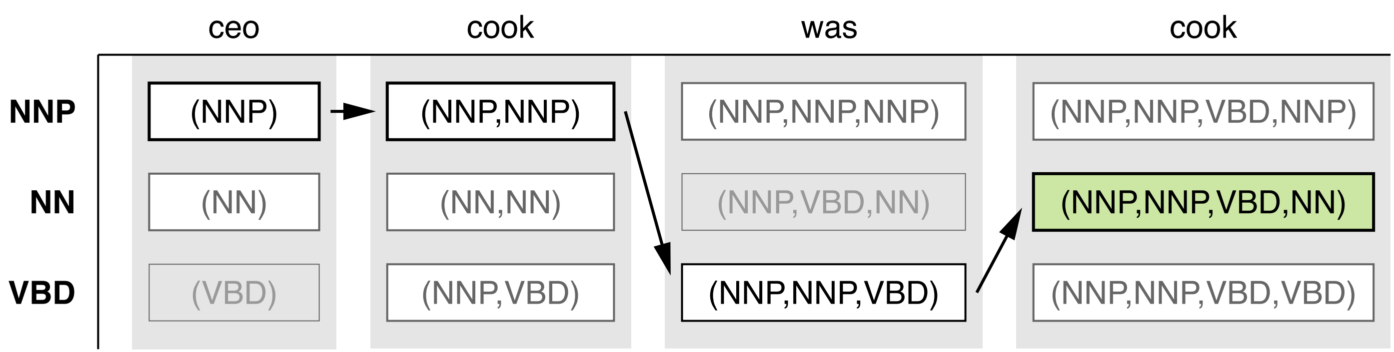
Discriminative models for sequence labeling

- Estimate the conditional probability $P(Y|O)$ of labels Y under spans O directly
- Often better on same data; focused on classification
- Example. Conditional random field



Decoding algorithm

- Computes the most likely sequence of labels for a given sequence of spans O based on a probabilistic sequence model
- Each span $o \in O$ is thereby assigned one label.
- Example. Viterbi algorithm



Evaluation of sequence labeling

- Classification. Accuracy over all instances (if all classes are important)
- Identification. Precision, recall, and F_1-score of BIO tags
- Identification+Classification. Micro and macro precision, recall, and F_1

Hidden Markov Models

Part-of-Speech (POS) Tagging

Part of speech (POS)

- A lexical category of a word, also called *word class*
- Abstract classes. Noun, verb, adjective, adverb, preposition, pronoun, ...

POS tags

- More fine-grained (partly language-specific) categories at token level
- The idea is to preserve more information that is easy to distinguish.

Common POS tag sets

- Penn Treebank. 45 tags for the English language (Marcus et al., 1993)
- Universal Dependencies. 17 tags for various languages (Nivre et al., 2016)

Selected Penn Treebank tags

Tag	Description	Example
VB	Verb base form	eat, be
VBG	Verb gerund	eating
VBP	Verb non-3sg pres.	eat, are
MD	Modal verb	can

Selected Universal Dependencies tags

Tag	Description	Example
VERB	Action or process	eat , eating
AUX	Helping verb marking	can, are
NOUN	Persons, things, ...	man, algorithm
PROPN	Names of persons, ...	Max, Viterbi

Part-of-Speech (POS) Tagging

POS Tagging

POS tagging

- Input. A sequence of tokens, usually those from one sentence
- Output. The POS tag of each token

> If/IN Tim/NNP Cook/NNP was/VBD a/DT cook/NN ,/, he/PRP would/MD cook/VB ./.

- Often used as a preprocessing step for other analyses
- The tag distribution and structure give insights into style and meaning.

Main challenges of POS tagging

- Ambiguity. Many words have multiple POS tags.
- Unknown words. Many words are not seen during training.

Example: Ambiguous POS tags

Apple's buildings are there in the back/NN.
A production facility is in the back/JJ part.
They bring back/RP production to the US.

Recently, they began to back/VB out of Asia.
US politicians back/VBP their decision.
This had been impossible back/RB then.

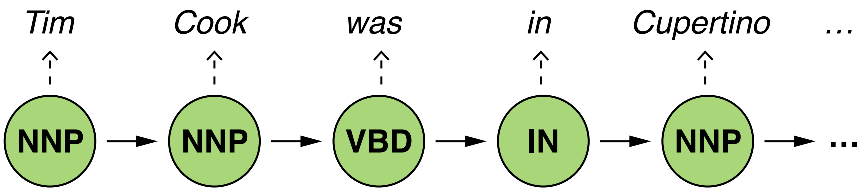
Hidden Markov Model (HMM)

Hidden Markov model (HMM)

- A probabilistic sequence model that defines the sequential distribution of a set of random variables $Q = \{q_1, \ldots, q_m\}$, called *states*
- States are not visible directly, but are predicted from *observations* o_i.

HMMs in NLP

- Observations. A sequence of text spans, such as words
- States. The possible labels of text spans, such as part-of-speech tags



HMMs for sequence labeling

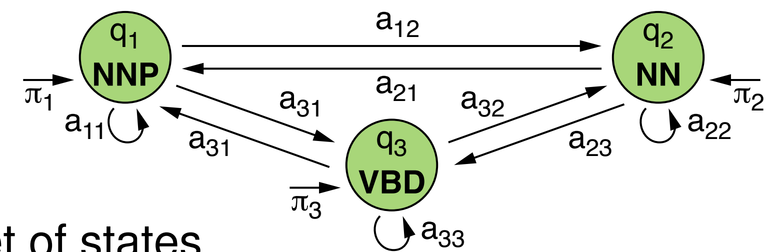
- Input. Any sequence of text spans $O = (o_1, \ldots, o_k)$
- Output. The most likely sequence of states $(q_1, \ldots, q_k)^*$ of Q
- HMMs model dependencies based on the *Markov assumption*.

Hidden Markov Model (HMM)

Markov Assumption and Markov Chains

Markov assumption

- Informally. When predicting the future, only the present matters.
- Fomally. Given a sequence of states $(q_1, \ldots, q_{i-1})$, the probability of the next state to be q_i is approximated as:
$$P(q_i \mid q_1, \ldots, q_{i-1}) \approx P(q_i \,|\, q_{i-1})$$
- The Markov assumption is embodied in a *Markov chain*.



Markov chain (Q, A, Π)

$Q \;:= \{q_i \mid 1 \leq i \leq m, m \in \mathbb{N}\}$ is a set of states,

$A \;:= \{a_{ij} \mid a_{ij} = P(q_j \,|\, q_i), 1 \leq i, j \leq m\}$ is a set of transition probabilities such that a_{ij} is the likelihood of q_j after q_i, $\forall i : \sum_{j=1}^{m} a_{ij} = 1$,

$\Pi \;:= \{\pi_i \mid 1 \leq i \leq m\}$ is a set of initial probabilities such that π_i is the likelihood to start in q_i, $\sum_{i=1}^{m} \pi_i = 1$.

Hidden Markov Model (HMM)

Definition

HMMs and Markov chains

- HMMs augment Markov chains, as states are observed indirectly only.
- States are assumed to be the causes of observations.

Hidden Markov model (Q, A, Π, O, B)

(Q, A, Π) is a Markov chain,

O $:= \{o_1, \ldots, o_n\}$ is a set of $n \geq 1$ observations,

B $:= \{b_{ij} \mid b_{ij} = P(o_j|q_i), 1 \leq i \leq m, 1 \leq j \leq n\}$ is a set of emission probabilities such that b_{ij} is the likelihood of observing o_j in q_i, $\forall j : \sum_{i=1}^{m} b_{ij} = 1$.

Observation independence assumption

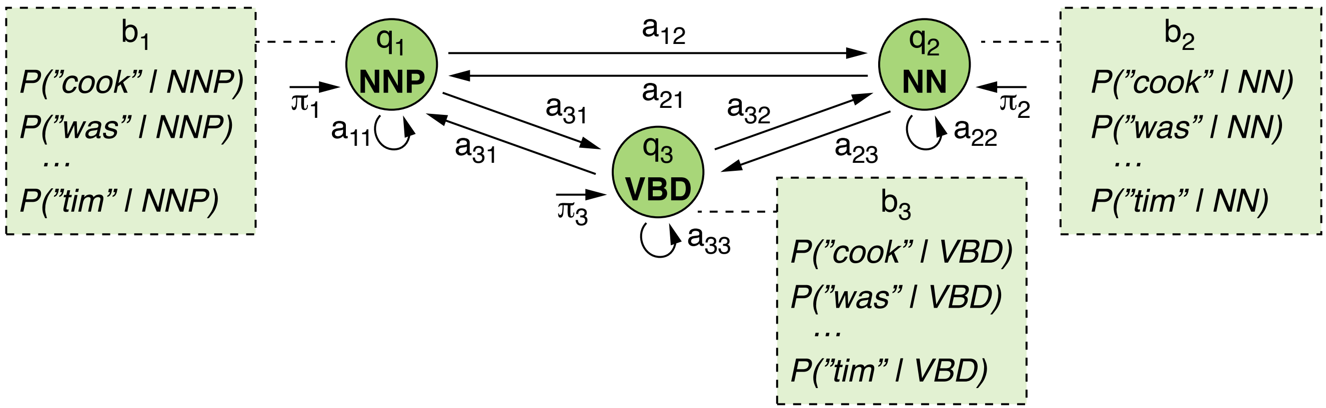
- In addition to the Markov assumption, the definition assumes that an observation o_i depends only on the current state q_i:

$$P(o_i \mid q_1, \ldots, q_i, o_1, \ldots, o_{i-1}) \approx P(o_i \mid q_i)$$

Hidden Markov Model (HMM)

Illustration and Estimation

Example: A toy HMM for POS tagging



Probability estimation in NLP

- Given a text corpus with class labels, all HMM probabilities can be estimated based on counts ($\#$) of labels y and spans o:
- Transitions. The probability of state q_{i+1} after q_i is estimated as:

$$P(q_{i+1} \mid q_i) = P(y_{i+1} \mid y_i) \approx \frac{\#(y_i, y_{i+1})}{\#y_i}$$

- Emissions. The probability of observation o_i in state q_i is estimated as:

$$P(o_i \mid q_i) = P(o_i \mid y_i) \approx \frac{\#(o_i, y_i)}{\#y_i}$$

Viterbi Algorithm

Decoding

- Input. A sequence of observations $(o_1, \ldots, o_k)$
- Output. The most likely sequence of states $(q_1, \ldots, q_k)^*$

Mathematical solution to decoding

- The goal is to determine:

$$(q_1, \ldots, q_k)^* \; := \underset{(q_1,\ldots,q_k) \in Q^k}{\operatorname{argmax}} P(q_1, \ldots, q_k \mid o_1, \ldots, o_k)$$

- Applying Bayes' rule:

$$\begin{aligned}(q_1, \ldots, q_k)^* &= \underset{(q_1,\ldots,q_k) \in Q^k}{\operatorname{argmax}} \frac{P(o_1, \ldots, o_k \mid q_1, \ldots, q_k) \cdot P(q_1, \ldots, q_k)}{P(o_1, \ldots, o_k)} \\ &= \underset{(q_1,\ldots,q_k) \in Q^k}{\operatorname{argmax}} P(o_1, \ldots, o_k \mid q_1, \ldots, q_k) \cdot P(q_1, \ldots, q_k)\end{aligned}$$

- Applying the two HMM assumptions: (let $P(q_1|q_0) := \pi_1$)

$$(q_1, \ldots, q_k)^* = \underset{(q_1,\ldots,q_k) \in Q^k}{\operatorname{argmax}} \prod_{i=1}^{k} P(o_i \mid q_i) \cdot P(q_i \mid q_{i-1})$$

Viterbi Algorithm

Decoding with the Viterbi algorithm

Viterbi algorithm in a nutshell

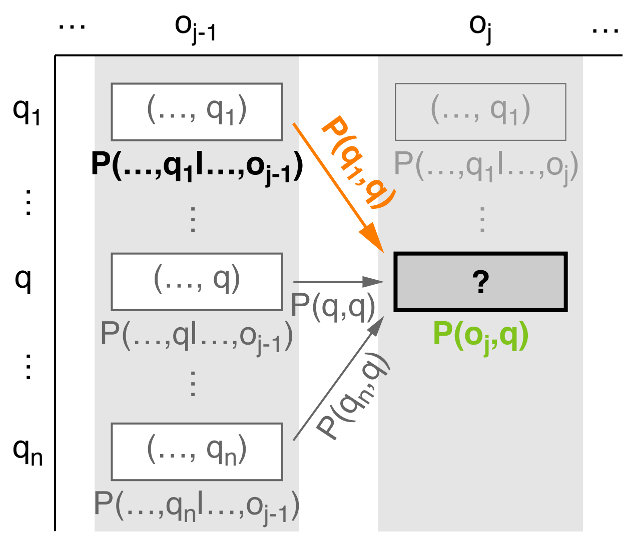
- A dynamic programming algorithm for the decoding problem
- For each observation sequence $(o_1, \ldots, o_j)$, $1 \leq j \leq k$, and state $q \in Q$, it computes the state sequence $(q_1, \ldots, q_j)^*$ that ends with $q_j = q$.
- The state sequence of length k with maximum probability is returned.

Viterbi probability computation

- Probability of $(q_1, \ldots, q)$ at o_j:

$$\underbrace{P(q_1, \ldots, q_{j-1} | o_1, \ldots, o_{j-1})}_{\text{Preceding sequence}} \cdot \underbrace{P(q | q_{j-1})}_{\text{Transition}} \cdot \underbrace{P(o_j | q)}_{\text{Emission}}$$

- This probability is computed once for each of the $|Q|$ preceding sequences.
- The one with highest probability is kept.



Viterbi Algorithm

Pseudocode

Signature

- Input. HMM (Q, A, Π, O, B) and a sequence of observations $(o_1, \ldots, o_k)$
- Output. The most probable sequence of states $(q_1, \ldots, q_k)^*$

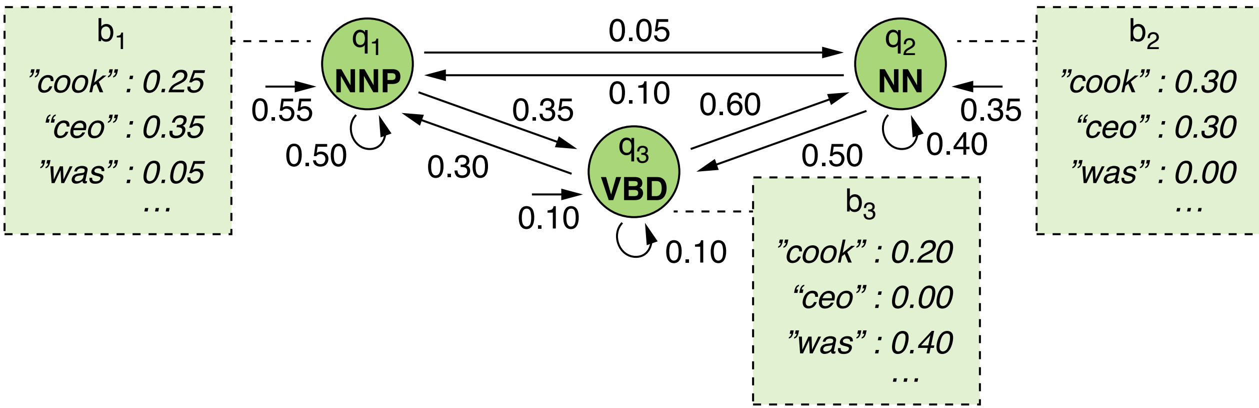
viterbi((Q, A, Π, O, B)**,** $(o_1, \ldots, o_k)$**)**

```
double [][] probs ← new double[|Q|][k]
List<State> [][] stateSeqs ← new List<State>[|Q|][k]
for each i ← 1 to |Q| do // Sequences of length 1
    probs[i][1] ← π_i · b_i1
    stateSeqs[i][1].append(q_i)
for each j ← 2 to k do // Sequences of length 2 to k
    for each i ← 1 to |Q| do
        probs[i][j] ← max_{l=1}^{|Q|} probs[l][j-1] · a_li · b_ij
        int best ← argmax_{l=1}^{|Q|} probs[l][j-1] · a_li · b_ij
        stateSeqs[i][j] ← stateSeqs[best][j-1].append(q_i)
int best ← argmax_{i=1}^{|Q|} probs[i][k]
return stateSeqs[best][k]
```

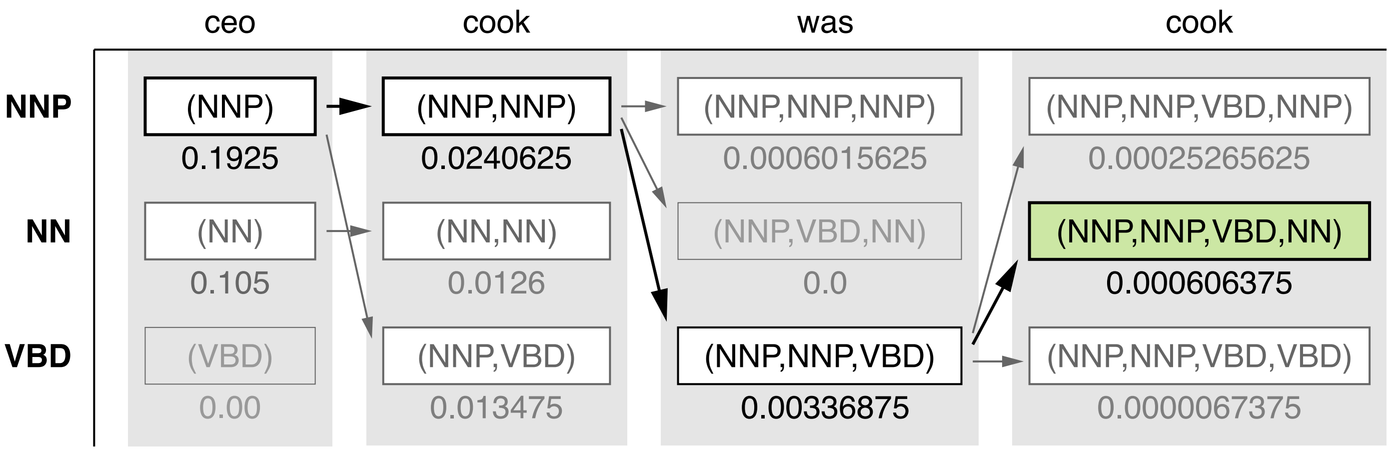
Viterbi Algorithm

Example

Toy HMM with concrete probabilities



Viterbi POS tagging as a lattice



Part-of-Speech (POS) Tagging

Evaluation of POS tagging

Data

- Wall Street Journal corpus, Penn Treebank tag set (Marcus et al., 1993)
- Types. 14% out of 51,457 different tokens ambiguous
- Instances. 55% out of 1,289,201 token occurrences ambiguous

Effectiveness results

- Majority baseline. For each token, predict its training majority tag.
- Human performance. Average of manual tagging through experts

Approach	**Accuracy**
Majority baseline	0.92
Human performance	**0.97**
HMM-based decoding	**0.97**

Notes

- Other methods perform equally well, such as CRFs or BERT classifiers.
- Similar effectiveness was observed on other English treebanks.
- POS tagging may be worse on less formal text and other languages.

Conditional Random Fields

Named Entity Recognition (NER)

Entity

- An entity represents a being or object from the real world.
- Entities can be seen as the basic semantic concept aimed for in NLP.

Named entity

- Any being or object that can be denoted with a proper name
- Both generic and domain-specific types of named entities exist.

Generic named entities

Type	Tag	Categories	Examples
Persons	PER	People, characters, ...	Tim Cook is Apple's CEO.
Location	LOC	Countries, regions, rivers, ...	He works in the Bay Area, California.
Organization	ORG	Companies, sports teams, ...	Apple brings back production to USA.
Geo-political	GPE	Countries, states, ...	The US passed a law supporting this.

Domain-specific named entities

- Any concept that is referred to with a name in a specific context
- Examples. Products, genes, works of art, ...

Named Entity Recognition (NER)

Example

Excerpt of a news article

[ChatGPT]$_{PRD}$: Why the human-like AI chatbot suddenly has everyone talking

By [Luke Hurst]$_{PER}$ - Updated: 15/12/2022

Long promised by science fiction, an artificial intelligence that can talk to you in natural language, and answer almost any questions you might have, is here. [ChatGPT]$_{PRD}$ has been taking social media by storm over the past week, with users showcasing the diverse ways the tool can be used. In just five days, it racked up over a million users, a feat that took social media platform [Meta]$_{ORG}$ (formerly [Facebook]$_{ORG}$) 10 months and streaming platform [Netflix]$_{ORG}$ three years to match.

Developed by the AI research company [OpenAI]$_{ORG}$, which has backers including [Microsoft]$_{ORG}$ and [Elon Musk]$_{PER}$, the chat tool uses the company's [GPT3]$_{PRD}$ ([Generative Pre-Trained Transformer 3]$_{PRD}$) technology to allow users to talk to the AI about almost anything. Trained on a massive data set, it is one of the most powerful language processing models ever created, and is able to respond in different styles [...]

Source: https://www.euronews.com/next/2022/12/14/chatgpt-why-the-human-like-ai-chatbot-suddenly-got-everyone-talking

Named Entity Recognition (NER)

NER

Named entity recognition (NER)

- Input. A sequence of tokens, usually those from a sentence or text
- Output. The spans that represent named entities of specific types
- Usually, NER is tackled as a BIO tagging problem.

Tim/B-PER Cook/I-PER works/O in/O Cupertino/B-LOC ./O

Why NER?

- Core analysis step in NLP for various downstream tasks
- Examples. Question answering, database population, knowledge linking

Main challenges of NER

- Sparseness. Often, only small portions of a text denote named entities.
- Unboundedness. Entity types are open word classes.
- Shape ambiguity. Named entities may just look like common nouns.
- Type ambiguity. Identical names are used for entities of different types.

Conditional Random Field (CRF)

Limitations of HMMs

- Much data is needed to learn a joint distribution $P(O, Q)$.
- No features other than sequence information can be exploited.
- Default HMMs cannot handle unknown spans o_u, since $\forall q : P(o_u|q) = 0$.

  Some workarounds exist, but they do not solve the limitations in general.

Conditional random field (CRF)

- A sequence model that estimates the probability of entire sequences of labels Y directly, given a sequence of text spans O
- Just like a supervised classifier, a CRF can integrate arbitrary features.
- We focus on *linear-chain CRFs*, which are used in NLP mostly.

CRFs vs. HMMs

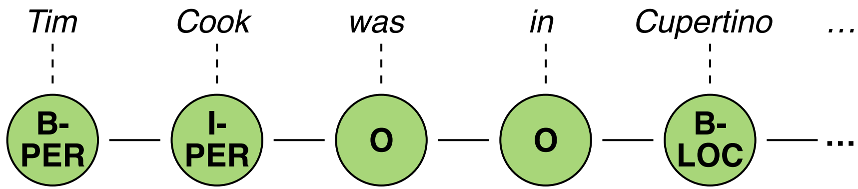
- The concepts of HMMs define the basis of CRFs.
- CRFs usually outperform HMMs on tasks where sequence information alone is rarely enough, such as NER.

Conditional Random Field (CRF)

Idea

CRFs for sequence labeling

- Input. Any sequence of text spans $O = (o_1, \ldots, o_k)$
- Output. The most likely sequence of labels $Y^* = (y_1, \ldots, y_k)^*$ under all possible sequences of labels $Y \in C^k$ of a set of labels C



Features in a CRF

- A CRF computes a set of *global features* over a set of *local features*.
- Local. Any measurable property f_{ij} of a span $o_j \in O$
  The number of local features depends on the number of spans k.
- Global. A function $F_i(O, Y)$ that sums up all values of k local features f_{ij}
  The set of m global features $F_1, \ldots, F_m$ is fixed.
- From the global features, the probability of any $Y \in C^k$ is predicted.

Conditional Random Field (CRF)

Probability Prediction in Linear-Chain CRFs

Linear-chain CRF

- A CRF where all f_{ij} depend only on the previous label y_{j-1}, the label y_j, the sequence of spans $O = (o_1, \ldots, o_k)$, and the position j:

$$F_i(O, Y) \ := \ \sum_{j=1}^{k} f_{ij}(y_{j-1}, y_j, O, j)$$

- This restriction enables the use of the Viterbi algorithm for decoding.

Probability prediction

- Given a sequence of spans $O = (o_1, \ldots, o_k)$, the probability of the sequence of labels $Y = (y_1, \ldots, y_k)$ is predicted as:

$$P(Y|O) \ := \ \exp\Big(\sum_{i=1}^{m} w_i \cdot F_i(O, Y)\Big) \Big/ \underbrace{\sum_{Y \in C^k} \exp\Big(\sum_{i=1}^{m} w_i \cdot F_i(O, Y)\Big)}_{\mathcal{F}(O)}$$

- The weights $w_1, \ldots, w_m$ of global features are learned on a training set.

Conditional Random Field (CRF)

Features in CRF-based POS tagging

Example features for POS tagging

- Tokens and tags (similar to HMM modeling)

$$
\begin{aligned}
f_{1,j}(y_{j-1}, y_j, O, j) &:= (\text{lower-case}(o_j) = \text{“cook”}, y_j = \text{NN})? \\
f_{2,j}(y_{j-1}, y_j, O, j) &:= (\text{lower-case}(o_j) = \text{“cook”}, y_j = \text{NNP}, o_{j+1} = \text{“cooked”})? \\
f_{3,j}(y_{j-1}, y_j, O, j) &:= (y_j = \text{“NNP”}, y_{j-1} = \text{NNP}, j = 2)?
\end{aligned}
$$

- Word shapes (particularly helpful for unknown words)

$$
\begin{aligned}
f_{101,j}(y_{j-1}, y_j, O, j) &:= (\text{abstract-letter-pattern}(o_j) = \text{Xx})? && \text{(e.g., “Cook”)} \\
f_{102,j}(y_{j-1}, y_j, O, j) &:= (\text{abstract-letter-pattern}(o_j) = \text{X})? && \text{(e.g., “CEO”)} \\
f_{103,j}(y_{j-1}, y_j, O, j) &:= (\text{suffix}(o_j) = \text{“ed”})? && \text{(e.g., “cooked”)}
\end{aligned}
$$

Example feature values

- Training example:

CEO/NNP Cook/NNP cooked/VBD ./.

- Feature values for $o_2 =$ “Cook”:

$f_{1,2} = 0,\ f_{2,2} = 1,\ f_{3,2} = 1$

$f_{101,2} = 1,\ f_{102,2} = 0,\ f_{103,2} = 0$

Conditional Random Field (CRF)

Features in CRF-based NER

Example features for NER

- Tokens and embeddings

Current token	Is token in person name list?	Current embedding
Previous token	Is token in location gazetteer?	Previous embedding
Following token	Is token in language lexicon?	Following embedding

- Style and word shapes

Current POS tag	Current abstract letter pattern	Current prefix
Previous POS tag	Previous abstract letter pattern	Current suffix
Following POS tag	Following abstract letter pattern	Character length

- Tags (no use of following tag!)

Current BIO tag	Current token-tag pair	Current tag, next token
Previous BIO tag	Previous token-tag pair	...

Notice

- The examples are feature *types*, many with several individual features.

Viterbi Algorithm for CRFs

Decoding

- Input. A sequence of spans $O = (o_1, \ldots, o_k)$
- Output. The most likely sequence of labels $Y^* = (y_1, \ldots, y_k)^*$

Mathematical solution to decoding

- Analog to HMMs, the goal is to determine:

$$(y_1, \ldots, y_k)^* \;:=\; \underset{(y_1,\ldots,y_k)\in C^k}{\operatorname{argmax}} P(y_1, \ldots, y_k \mid o_1, \ldots, o_k)$$

- Using the probability prediction from above:

$$\begin{aligned}(y_1, \ldots, y_k)^* \;&=\; \underset{(y_1,\ldots,y_k)\in C^k}{\operatorname{argmax}} \exp\left(\sum_{i=1}^{m} w_i \cdot F_i(O, Y)\right) \Big/ \mathcal{F}(O)\\ &=\; \underset{(y_1,\ldots,y_k)\in C^k}{\operatorname{argmax}} \exp\left(\sum_{i=1}^{m} w_i \cdot \sum_{j=1}^{k} f_{ij}(y_{j-1}, y_j, O, j)\right) \Big/ \mathcal{F}(O)\end{aligned}$$

- Neither $\exp$ nor $\mathcal{F}(O)$ change the argmax:

$$(y_1, \ldots, y_k)^* \;=\; \underset{(y_1,\ldots,y_k)\in C^k}{\operatorname{argmax}} \sum_{j=1}^{k}\sum_{i=1}^{m} w_i \cdot f_{ij}(y_{j-1}, y_j, O, j)$$

Viterbi Algorithm for CRFs

Decoding with the Viterbi algorithm

CRF decoding with the Viterbi algorithm

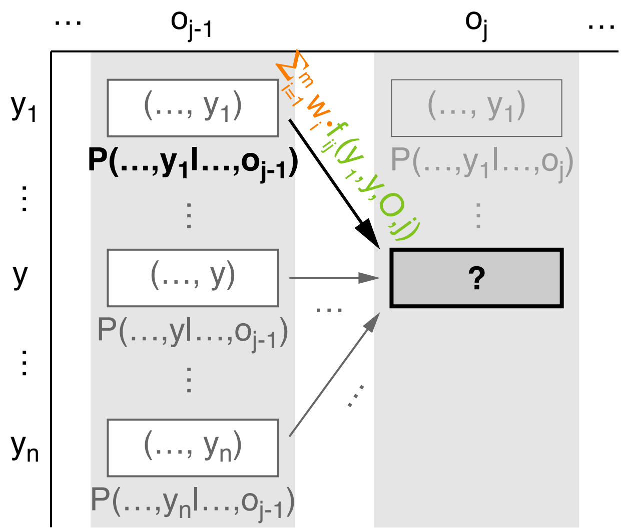
- For linear-chain CRFs, the Viterbi algorithm can be used again.
- Instead of probabilities, it predicts scores based on the features.
- While the scores of entire label sequences are predicted, this may also be done step by step for increasing sequence lengths j.

Score computation

- Predicted score of $(y_1, \ldots, y)$ at o_j:

$$\underbrace{P(y_1, \ldots, y_{j-1} | o_1, \ldots, o_{j-1})}_{\text{Preceding sequence}} \cdot \underbrace{\sum_{i=1}^{m} w_i \cdot f_{ij}(y_{j-1}, y, O, j)}_{\text{Prediction at } j-1 \text{ to } j}$$

- This score is computed once for each label in C.
- The one with highest score is kept.



Viterbi Algorithm for CRFs

Pseudocode

Signature

- Input. An array of n labels C, and a sequence of spans $(o_1, \ldots, o_k)$
- Output. The most likely sequence of labels $(y_1, \ldots, y_k)^*$

```
viterbiCRF(String [] C, List<String> (o_1,...,o_k))
    double [][] scores ← new double[n][k]
    List<String> [][] labelSeqs ← new List<String>[n][k]
    for each r ← 1 to n do
        scores[r][1] ← Σ_{i=1}^{m} w_i · f_ij(⊥,C[r],O,j) // No prev. label
        labelSeqs[r][1].append(C[r])
    for each j ← 2 to k do
        for each r ← 1 to n do
            scores[r][j] ← max_{l=1}^{n} scores[l][j-1]
                · Σ_{i=1}^{m} w_i · f_ij(labelSeqs[l][j-1].last(),C[r],O,j)
            int best ← argmax_{l=1}^{n} scores[l][j-1]
                · Σ_{i=1}^{m} w_i · f_ij(labelSeqs[l][j-1].last(),C[r],O,j)
            labelSeqs[r][j] ← labelSeqs[best][j-1].append(y_r)
    int best ← argmax_{r=1}^{n} scores[r][k]
    return labelSeqs[best][k]
```

Named Entity Recognition (NER)

Evaluation of NER

Data (Tjong Kim Sang and De Meulder, 2003)

- CoNLL-2003 shared task dataset with 1393 English news articles
- Named entities. PER (10048), LOC (10645), ORG (9323), MISC (5062)
  About 2/3 for training, 1/6 validation, 1/6 test

Selected effectiveness results

Approach	Micro F_1
HMM with feature extension (Florian et al., 2003)	0.820
SVM with rich boundary features (Li et al., 2005)	0.863
HMM + other classifiers (Florian et al., 2003)	0.888
CRF with embedding features (Passos et al., 2014)	0.909
CRF + recurrent neural network (Jiang et al., 2019)	0.935
Transformers + reinforcement learning (Wang et al., 2021)	**0.946**

Observation

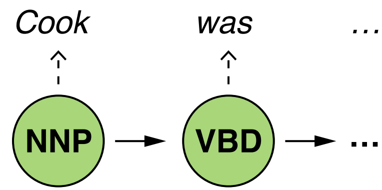
- CRFs are almost as good as neural methods and used as part of them.

Conclusion

Conclusion

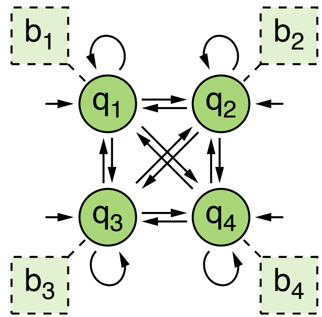
NLP using sequence labeling

- Prediction of most likely sequences of class labels
- Either span-by-span or jointly for a span sequence
- Key technique for any span-based labeling task



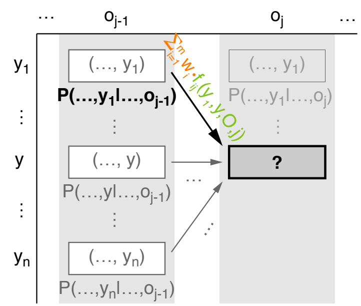
Hidden markov model (HMM)

- Generative model for sequence labeling
- Most likely sequence found by Viterbi algorithm
- Fundamental method for POS tagging and similar



Conditional random field (CRF)

- Discriminative model for sequence labeling
- Linear-chain CRFs work with Viterbi algorithm, too
- Standard method for NER and similar



References

Much content and examples based on

- **Jurafsky and Martin (2021).** Daniel Jurafsky and James H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. Draft or 3rd edition, December 29, 2021. https://web.stanford.edu/ jurafsky/slp3/

Other references

- **Florian et al. (2003).** Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named Entity Recognition through Classifier Combination. In In Proceedings of the Seventh Conference on Natural Language Learning, pages 168–171, 2003.
- **Jian et al. (2019).** Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, Jingbo Zhu. Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3585–3590, 2019.
- **Li et al. (2005).** Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. 2005. SVM based Learning System for Information Extraction. In Deterministic and Statistical Methods in Machine Learning, pages 319–339. Springer.
- **Marcus et al. (1993).** Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. Computational Linguistics, 19(2):313–330, 1993.

References

Other references

- **Nivre et al. (2016).** Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal Dependencies v1: A multilingual tree- bank collection. LREC, pages 1659–1666, 2016.
- **Passos et al. (2014).** Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon Infused Phrase Embeddings for Named Entity Resolution. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning, pages 78–86, 2014.
- **Ramshaw and Marcus (1995).** Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking using Transformation-based Learning. In Proceedings of the Third Workshop on Very Large Corpora, pages 82–94, 1995.
- **Tjong Kim Sang and Fien De Meulder (2003).** Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural language Learning at HLT-NAACL 2003 - Volume 4, pages 142–147, 2003.
- **Wang et al. (2021).** Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. Automated Concatenation of Embeddings for Structured Prediction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2643–2660, 2021.