-->

Chapter 5: Information Extraction — Sequence Labeling and Structured Prediction

Covered Material	Value
Chapters	Chapters 17 and 20 (Jurafsky & Martin)
Related Week	Week 5
Related Slides / Files	week5.md, 17.md, 20.md

Index

Brief Content Summary

This chapter addresses the transition from document-level classification to sequence labeling, where the goal is to assign a label to each element in a sequence rather than a single label to an entire input. Part-of-speech (POS) tagging and Named Entity Recognition (NER) serve as canonical examples of sequence labeling tasks. The chapter develops Hidden Markov Models (HMMs) as generative sequence models, deriving the transition and emission probability estimation via Maximum Likelihood Estimation (MLE). The Viterbi algorithm is presented as a dynamic programming solution for efficient decoding. Conditional Random Fields (CRFs) are introduced as discriminative alternatives that allow arbitrary feature functions while maintaining tractable inference. The chapter extends to information extraction tasks including relation extraction via patterns, supervised learning, distant supervision, and open information extraction. Temporal and event extraction complete the treatment, introducing Allen's interval algebra for temporal reasoning.

Abstract

This chapter provides a rigorous treatment of sequence labeling and information extraction in Natural Language Processing (NLP). The exposition begins with the sequence labeling paradigm, where each token in an input sequence receives a corresponding label, contrasted with global classification approaches. Part-of-speech tagging and Named Entity Recognition are formalized as sequence labeling tasks, with BIO tagging schemes enabling span detection. Hidden Markov Models are developed as generative probabilistic models that jointly model observed sequences and hidden state sequences through transition and emission probabilities estimated via Maximum Likelihood. The Viterbi algorithm is derived as a dynamic programming solution to the decoding problem, enabling efficient computation of the most

probable state sequence in $O(TN^2)$ time. Conditional Random Fields are introduced as log-linear discriminative sequence models that directly model the posterior probability of label sequences given observations, permitting rich feature engineering. The chapter then addresses information extraction: relation extraction via Hearst patterns, supervised classification, bootstrapping, distant supervision, and unsupervised Open IE methods. Temporal analysis including event detection, temporal expression normalization, and Allen's interval algebra concludes the treatment.

Keywords

Sequence Labeling; Part-of-Speech Tagging; Named Entity Recognition; BIO Tagging; Hidden Markov Models; Markov Assumption; Transition Probabilities; Emission Probabilities; Viterbi Algorithm; Dynamic Programming; Conditional Random Fields; Feature Functions; Relation Extraction; Hearst Patterns; Distant Supervision; Open Information Extraction; Temporal Expressions; Allen's Interval Algebra

---

1 Introduction

This section motivates the transition from global classification to sequence labeling and provides an overview of the chapter.

1.1 From Global to Sequential Classification

Previous chapters have addressed classification tasks where the goal is to assign a single label to an entire document or sentence—sentiment classification, topic categorization, or spam detection. However, many NLP tasks require finer-grained predictions: assigning a label to each token in a sequence.

Consider the sentence:

> *United Airlines said Friday it has increased fares.*

A global classifier might label this sentence as "business news." But for many applications, we need token-level annotations:

- **Part-of-speech tagging**: United/NNP Airlines/NNP said/VBD Friday/NNP it/PRP has/VBZ increased/VBN fares/NNS
- **Named entity recognition**: [ORG United Airlines] said [TIME Friday] it has increased fares

These tasks share a common structure: given an input sequence $X = x_1, x_2, \ldots, x_n$, predict an output sequence $Y = y_1, y_2, \ldots, y_n$ of the same length, where each y_i is drawn from a finite label set.

The key insight motivating sequence models is that labels exhibit **dependencies**. The probability of a word being a verb depends on the preceding word's part of speech; the probability of a token being inside a named entity depends on whether the previous token began that entity. Treating each token independently ignores these sequential dependencies and degrades performance.

1.2 Chapter Overview

The chapter proceeds as follows:

- **Section 2** introduces parts of speech and the POS tagging task.
- **Section 3** formalizes Named Entity Recognition with BIO tagging.
- **Sections 4–5** develop Hidden Markov Models and the Viterbi algorithm.
- **Section 6** presents Conditional Random Fields as discriminative alternatives.
- **Sections 7–8** extend to information extraction: relations, events, and temporal expressions.

---

2 Parts of Speech and Word Classes

This section introduces the linguistic categories that form the label set for POS tagging.

2.1 Open and Closed Class Words

Parts of speech divide into two broad categories based on productivity:

Closed class words belong to categories with relatively fixed membership. New prepositions, determiners, or pronouns are rarely coined. These words are typically short, frequent, and serve grammatical (functional) rather than content-bearing roles. Examples include:

- Determiners: *the*, *a*, *this*

- Prepositions: *in*, *on*, *by*
- Pronouns: *he*, *she*, *it*
- Conjunctions: *and*, *or*, *but*

Open class words belong to categories that readily accept new members through coinage, borrowing, or derivation. Examples include:

- Nouns: *algorithm*, *iPhone*, *COVID*
- Verbs: *google*, *tweet*, *zoom*
- Adjectives: *viral*, *sustainable*
- Adverbs: *basically*, *literally*

The distinction matters computationally: closed-class words are well-covered by any reasonable lexicon, while open-class words include many rare or previously unseen items (unknown words).

2.2 Part-of-Speech Tagsets

A **tagset** defines the inventory of part-of-speech labels. Two widely used tagsets are:

The Universal Dependencies (UD) Tagset comprises 17 tags designed for cross-linguistic applicability:

Tag	Description	Examples
ADJ	Adjective	*red*, *young*
ADV	Adverb	*slowly*, *very*
NOUN	Common noun	*cat*, *algorithm*
VERB	Verb	*run*, *compute*
PROPN	Proper noun	*Stanford*, *Marie*
DET	Determiner	*the*, *a*
ADP	Adposition	*in*, *on*
AUX	Auxiliary	*can*, *will*
PRON	Pronoun	*she*, *they*
NUM	Numeral	*one*, *2024*

The Penn Treebank Tagset comprises 36–45 tags with finer distinctions for English:

Tag	Description	Example
NN	Singular noun	*cat*
NNS	Plural noun	*cats*
NNP	Singular proper noun	*London*
VB	Verb base form	*eat*
VBD	Verb past tense	*ate*
VBG	Verb gerund	*eating*
VBN	Verb past participle	*eaten*
JJ	Adjective	*happy*
RB	Adverb	*quickly*
MD	Modal	*can*, *should*

2.3 Ambiguity in Tagging

POS tagging is a **disambiguation** task. While approximately 85% of word *types* in English are unambiguous (appearing with only one tag in corpora), these unambiguous types account for only 33–45% of word *tokens* in running text. The remaining 55–67% of tokens are ambiguous.

Common ambiguous words include:

Word	Possible Tags	Examples
back	JJ, NN, VB, VBP, RP, RB	*back seat* (JJ), *in the back* (NN), *back the bill* (VB)
that	DT, IN	*that book* (DT), *I know that...* (IN)
will	MD, NN	*will go* (MD), *last will* (NN)

Most Frequent Class Baseline. A simple baseline assigns each word its most frequent tag from training data. This baseline achieves approximately 92% accuracy—demonstrating both that the task is highly constrained and that the remaining 5% improvement (to reach state-of-the-art ~97%) requires modeling sequential dependencies.

---

3 Named Entity Recognition

This section formalizes the Named Entity Recognition task and introduces span-based tagging schemes.

3.1 Named Entity Types

Named Entity Recognition (NER) is the task of identifying spans of text that refer to specific entities and classifying them into predefined categories. Unlike POS tagging, which labels individual tokens, NER identifies contiguous spans of potentially multiple tokens.

Standard entity types include:

Type	Description	Examples
PER	Person names	*Marie Curie*, *Barack Obama*
ORG	Organizations	*Google*, *United Nations*
LOC	Locations	*Paris*, *Mount Everest*
GPE	Geo-political entities	*France*, *California*
TIME	Temporal expressions	*Friday*, *January 2024*
MONEY	Monetary values	*$50*, *€100 million*

The choice of entity types depends on the application domain. Biomedical NER uses types such as GENE, PROTEIN, DISEASE, and DRUG. Legal NER identifies CASE, COURT, and STATUTE entities.

3.2 BIO Tagging Schemes

NER poses a span identification problem: how do we represent multi-token entities using per-token labels? The **BIO tagging scheme** solves this by introducing prefix tags:

- **B-X**: Beginning of an entity of type X
- **I-X**: Inside (continuation of) an entity of type X
- **O**: Outside any entity

Consider the sentence:

> *United Airlines said Friday it has increased fares.*

The BIO-tagged representation:

Token	BIO Tag
United	B-ORG
Airlines	I-ORG
said	O

Token	BIO Tag
Friday	B-TIME
it	O
has	O
increased	O
fares	O

This encoding reduces NER to sequence labeling: given n tokens, predict n BIO tags. The key insight is that the B- prefix marks entity boundaries, enabling recovery of multi-token spans.

BIOES/BIOLU Tagging. An extension uses additional prefixes:

- **B**: Beginning of a multi-token entity
- **I**: Inside a multi-token entity
- **O**: Outside any entity
- **E** (or **L**): End of a multi-token entity
- **S** (or **U**): Singleton (single-token entity)

The BIOES scheme provides stronger boundary signals, often improving model performance:

Token	BIOES Tag
United	B-ORG
Airlines	E-ORG
said	O
Friday	S-TIME

3.3 Challenges in NER

Several factors complicate NER:

Nested entities. Some entities contain other entities:

> *[ORG Bank of [GPE America]]*

Standard sequence labeling cannot represent nested structures; each token receives exactly one tag.

Metonymy. Location names may refer to organizations:

> *Paris announced new climate targets.*

Here "Paris" refers to the French government, not the geographic location.

Unknown entities. Many named entities (especially person and organization names) are rare or novel. Unlike closed-class words, entities exhibit a Zipfian distribution with many singletons.

Context dependence. The same string may denote different entity types:

> *Washington crossed the Delaware.* (PER)
> *Washington is the capital.* (GPE)

---

4 Hidden Markov Models

This section develops Hidden Markov Models as generative probabilistic models for sequence labeling.

4.1 Markov Chains

A **Markov chain** is a stochastic process over a sequence of states $q_1, q_2, \ldots, q_T$ satisfying the **Markov assumption**: the probability of transitioning to a state depends only on the current state, not the entire history.

First-Order Markov Assumption:

$$P(q_i|q_1, q_2, \ldots, q_{i-1}) = P(q_i|q_{i-1})$$

This memoryless property enables tractable computation: instead of conditioning on exponentially many histories, we condition only on the immediate predecessor.

A Markov chain is specified by:

- A set of states $Q = \{q_1, q_2, \ldots, q_N\}$
- A transition probability matrix $A = [a_{ij}]$ where $a_{ij} = P(q_j|q_i)$
- An initial probability distribution $\pi = [\pi_i]$ where $\pi_i = P(q_1 = i)$

The transition probabilities satisfy:

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$$

4.2 The Hidden Markov Model

An **Hidden Markov Model (HMM)** extends the Markov chain by introducing **observations**. The states are now *hidden* (unobserved), and we observe only emissions from each state.

Formally, an HMM is a 5-tuple $\lambda = (Q, A, O, B, \pi)$:

Component	Definition
Q	Set of N hidden states
A	Transition probability matrix, $A = [a_{ij}]$
O	Set of M observation symbols
B	Emission probability matrix, $B = [b_i(o_k)]$
π	Initial state distribution

For POS tagging:

- Hidden states Q are POS tags (NN, VB, JJ, ...)
- Observations O are words
- Transitions A capture tag-to-tag dependencies
- Emissions B capture tag-to-word generation

The model makes two independence assumptions:

Markov Assumption (transitions):

$$P(q_i|q_1, \ldots, q_{i-1}) = P(q_i|q_{i-1})$$

Output Independence (emissions):

$$P(o_i|q_1, \ldots, q_T, o_1, \ldots, o_T) = P(o_i|q_i)$$

The output independence assumption states that the emission at time i depends only on the current hidden state—not on other states or emissions.

4.3 HMM Components: Transition and Emission Probabilities

Transition Probabilities capture dependencies between consecutive hidden states. For POS tagging, $P(t_i|t_{i-1})$ models the likelihood of tag t_i following tag t_{i-1}.

The transition probability matrix A is typically augmented with special start ($\langle s \rangle$) and end ($\langle /s \rangle$) symbols:

$$A = \begin{bmatrix} P(\text{DET}|\langle s \rangle) & P(\text{NN}|\langle s \rangle) & P(\text{VB}|\langle s \rangle) & \cdots \\ P(\text{DET}|\text{DET}) & P(\text{NN}|\text{DET}) & P(\text{VB}|\text{DET}) & \cdots \\ P(\text{DET}|\text{NN}) & P(\text{NN}|\text{NN}) & P(\text{VB}|\text{NN}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Emission Probabilities capture the likelihood of generating a specific word given a hidden state. For POS tagging, $P(w_i|t_i)$ models the probability of observing word w_i when the tag is t_i.

The emission matrix B has one row per tag:

$$B = \begin{bmatrix} P(\text{the}|\text{DET}) & P(\text{a}|\text{DET}) & P(\text{cat}|\text{DET}) & \cdots \\ P(\text{the}|\text{NN}) & P(\text{a}|\text{NN}) & P(\text{cat}|\text{NN}) & \cdots \\ P(\text{the}|\text{VB}) & P(\text{a}|\text{VB}) & P(\text{cat}|\text{VB}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

4.4 Maximum Likelihood Estimation for HMMs

Given a labeled training corpus with word-tag pairs, the HMM parameters are estimated via **Maximum Likelihood Estimation (MLE)** using relative frequency counts.

Transition Probability Estimation:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

where $C(t_{i-1}, t_i)$ is the count of tag bigram (t_{i-1}, t_i) and $C(t_{i-1})$ is the count of tag t_{i-1}.

Emission Probability Estimation:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

where $C(t_i, w_i)$ is the count of word w_i tagged as t_i and $C(t_i)$ is the total count of tag t_i.

Example. Consider a corpus with the following counts:

Tag Bigram	Count
$\langle s \rangle$ DET	5000
DET NN	4500
NN VB	3200
VB DET	2100

Word-Tag	Count
(the, DET)	4800
(DET total)	5000
(cat, NN)	120
(NN total)	8000

Then:

$$P(\text{NN}|\text{DET}) = \frac{C(\text{DET}, \text{NN})}{C(\text{DET})} = \frac{4500}{5000} = 0.90$$

$$P(\text{the}|\text{DET}) = \frac{C(\text{DET}, \text{the})}{C(\text{DET})} = \frac{4800}{5000} = 0.96$$

$$P(\text{cat}|\text{NN}) = \frac{C(\text{NN}, \text{cat})}{C(\text{NN})} = \frac{120}{8000} = 0.015$$

Smoothing. Raw MLE assigns zero probability to unseen events. For unknown words, one approach uses an open class prior that distributes probability mass across likely tags (NN, VB, JJ, NNP) rather than closed-class tags (DET, IN).

---

5 The Viterbi Algorithm

This section presents the Viterbi algorithm for efficient HMM decoding.

5.1 Decoding as Optimization

The **decoding problem** asks: given an HMM λ and an observation sequence $O = o_1, o_2, \ldots, o_T$, find the most probable hidden state sequence $Q^* = q_1, q_2, \ldots, q_T$.

Applying Bayes' theorem:

$$\hat{t}_{1:n} = \arg\max_{t_{1:n}} P(t_{1:n}|w_{1:n})$$

Using the HMM assumptions, this simplifies to:

$$\hat{t}_{1:n} = \arg\max_{t_{1:n}} \prod_{i=1}^{n} P(w_i|t_i) \cdot P(t_i|t_{i-1})$$

where $P(w_i|t_i)$ is the emission probability and $P(t_i|t_{i-1})$ is the transition probability.

The Combinatorial Problem. A naive approach enumerates all possible tag sequences. For a tagset of size N and sequence length T, there are N^T possible sequences—exponential in sequence length. For $N = 36$ (Penn Treebank) and $T = 20$ (a typical sentence), this yields $36^{20} \approx 10^{31}$ sequences.

5.2 Dynamic Programming Solution

The **Viterbi algorithm** solves decoding in $O(TN^2)$ time using dynamic programming. The key insight is **optimal substructure**: the best path to a state at time t must pass through the best path to some state at time $t-1$.

The algorithm constructs a **lattice** (trellis) where:

- Columns represent time steps $t = 1, 2, \ldots, T$
- Rows represent hidden states $q_1, q_2, \ldots, q_N$
- Each cell (t, j) stores the probability of the best path ending in state j at time t

5.3 The Viterbi Recurrence

The algorithm maintains two quantities:

Viterbi path probability $v_t(j)$: the probability of the most probable path ending in state j at time t:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$$

where:

- $v_{t-1}(i)$ is the best path probability to state i at time $t-1$
- $a_{ij} = P(q_t = j | q_{t-1} = i)$ is the transition probability
- $b_j(o_t) = P(o_t | q_t = j)$ is the emission probability

Backpointer $\text{bt}_t(j)$: the state at time $t-1$ that maximizes the path probability to state j at time t:

$$\text{bt}_t(j) = \arg\max_{i=1}^{N} v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$$

Initialization ($t = 1$):

$$v_1(j) = \pi_j \cdot b_j(o_1)$$

where π_j is the initial probability of state j.

Termination:

$$P^* = \max_{i=1}^{N} v_T(i) \cdot a_{i,\text{END}}$$

The best final state is:

$$q_T^* = \arg\max_{i=1}^{N} v_T(i) \cdot a_{i,\text{END}}$$

Backtracking: Trace backpointers from q_T^* to recover the full state sequence.

5.4 Worked Example

Consider the sentence: "*the cat sat*" with tagset $\{$DET, NN, VB$\}$.

Transition probabilities A:

	DET	NN	VB
$\langle s \rangle$	0.8	0.1	0.1
DET	0.1	0.8	0.1
NN	0.1	0.2	0.7
VB	0.4	0.3	0.3

Emission probabilities B:

	the	cat	sat
DET	0.9	0.01	0.01
NN	0.01	0.7	0.1
VB	0.01	0.1	0.8

Step 1: Initialization ($t = 1$, word = "the")

$$v_1(\text{DET}) = P(\text{DET}|\langle s \rangle) \cdot P(\text{the}|\text{DET}) = 0.8 \times 0.9 = 0.72$$

$$v_1(\text{NN}) = 0.1 \times 0.01 = 0.001$$

$$v_1(\text{VB}) = 0.1 \times 0.01 = 0.001$$

Step 2: Recursion ($t = 2$, word = "cat")

For each state j, compute $v_2(j) = \max_i v_1(i) \cdot a_{ij} \cdot b_j(\text{cat})$:

$$v_2(\text{NN}) = \max \begin{cases} v_1(\text{DET}) \cdot a_{\text{DET,NN}} \cdot b_{\text{NN}}(\text{cat}) = 0.72 \times 0.8 \times 0.7 = 0.403 \\ v_1(\text{NN}) \cdot a_{\text{NN,NN}} \cdot b_{\text{NN}}(\text{cat}) = 0.001 \times 0.2 \times 0.7 = 0.00014 \\ v_1(\text{VB}) \cdot a_{\text{VB,NN}} \cdot b_{\text{NN}}(\text{cat}) = 0.001 \times 0.3 \times 0.7 = 0.00021 \end{cases}$$

Maximum: $v_2(\text{NN}) = 0.403$, backpointer: $\text{bt}_2(\text{NN}) = \text{DET}$

Step 3: Recursion ($t = 3$, word = "sat")

$$v_3(\text{VB}) = \max \begin{cases} v_2(\text{DET}) \cdot a_{\text{DET,VB}} \cdot b_{\text{VB}}(\text{sat}) \\ v_2(\text{NN}) \cdot a_{\text{NN,VB}} \cdot b_{\text{VB}}(\text{sat}) = 0.403 \times 0.7 \times 0.8 = 0.226 \\ v_2(\text{VB}) \cdot a_{\text{VB,VB}} \cdot b_{\text{VB}}(\text{sat}) \end{cases}$$

Maximum path: $v_3(\text{VB}) = 0.226$, backpointer: $\text{bt}_3(\text{VB}) = \text{NN}$

Backtracking: VB ← NN ← DET

Final result: DET NN VB → *the/DET cat/NN sat/VB*

---

6 Conditional Random Fields

This section introduces Conditional Random Fields as discriminative sequence models.

6.1 Limitations of Generative Models

HMMs are **generative models**: they model the joint probability $P(X, Y)$ of observations and labels. Decoding uses Bayes' rule:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

This generative approach has limitations:

1. **Independence assumptions.** HMMs assume output independence—that each word depends only on its tag. This prevents using features like neighboring words, word shape, or prefix/suffix information.

2. **Feature engineering difficulty.** Adding features to generative models requires modeling their distributions, potentially leading to complex and computationally expensive models.

3. **Explaining away.** Generative models must explain all variation in the observed data, even aspects irrelevant to the labeling task.

Discriminative models directly model $P(Y|X)$, avoiding explicit modeling of $P(X)$. This permits rich, overlapping, non-independent features.

6.2 CRF Formulation

A **Linear-Chain Conditional Random Field (CRF)** is a log-linear model for sequence labeling that directly estimates:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X, Y)\right)$$

where:

- $Y = y_1, y_2, \ldots, y_n$ is the label sequence
- $X = x_1, x_2, \ldots, x_n$ is the observation sequence
- $F_k(X, Y)$ are **global feature functions** summing over the sequence
- w_k are learned weights
- $Z(X)$ is the **partition function** ensuring normalization

The partition function sums over all possible label sequences:

$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X, Y')\right)$$

where $\mathcal{Y}$ is the set of all possible label sequences of length n.

6.3 Feature Functions

CRF features are defined locally and summed to obtain global features.

Local feature functions $f_k(y_{i-1}, y_i, X, i)$ examine:

- The previous label y_{i-1}
- The current label y_i
- The entire input sequence X
- The current position i

Global feature functions aggregate local features:

$$F_k(X, Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

Example Features for NER:

Feature Template	Description
$f(y_i = \text{ORG}, x_i = \text{"Inc."})$	Current word is "Inc." and label is ORG
$f(y_{i-1} = \text{B-PER}, y_i = \text{I-PER})$	Transition from B-PER to I-PER
$f(y_i = \text{PER}, \text{IsCapitalized}(x_i))$	Word is capitalized and label is PER
$f(y_i = \text{LOC}, x_{i-1} = \text{"in"})$	Previous word is "in" and label is LOC
$f(y_i = \text{PER}, \text{suffix}(x_i) = \text{"-son"})$	Word ends in "-son" and label is PER

Word Shape Features. Word shape abstracts orthographic patterns:

Word	Shape
McDonald's	XxXxxx'x
IBM	XXX

Word	Shape
42	dd
3.14	d.dd
U.S.A.	X.X.X.

Feature Templates. Rather than enumerating features, templates automatically generate features:

Template: `currentWord = X AND label = Y`

Applied to training data, this generates features for every (word, label) pair observed:

- currentWord=*the* AND label=O
- currentWord=*United* AND label=B-ORG
- currentWord=*Airlines* AND label=I-ORG
- ...

6.4 Inference with Viterbi

CRF decoding finds the label sequence maximizing $P(Y|X)$:

$$\hat{Y} = \arg\max_{Y} P(Y|X) = \arg\max_{Y} \sum_{k=1}^{K} w_k F_k(X, Y)$$

Since the exponential and partition function are monotonic, we maximize the unexponentiated score.

The **CRF Viterbi algorithm** adapts HMM Viterbi by replacing probabilities with weighted feature sums:

$$v_t(j) = \max_{i=1}^{N} \left[v_{t-1}(i) + \sum_{k=1}^{K} w_k f_k(y_i, y_j, X, t) \right]$$

Key differences from HMM Viterbi:

- **Addition** replaces multiplication (log-linear model)
- Features can examine the **entire input sequence** X, not just the current observation
- **Arbitrary features** beyond transition and emission templates

6.5 Training CRFs

CRF training maximizes conditional log-likelihood:

$$\mathcal{L}(\mathbf{w}) = \sum_{(X,Y) \in \text{Train}} \log P(Y|X; \mathbf{w})$$

Expanding:

$$\mathcal{L}(\mathbf{w}) = \sum_{(X,Y)} \left[\sum_k w_k F_k(X, Y) - \log Z(X) \right]$$

The gradient with respect to weight w_k:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \sum_{(X,Y)} \left[F_k(X, Y) - \mathbb{E}_{Y' \sim P(Y'|X)}[F_k(X, Y')] \right]$$

This gradient has an intuitive interpretation:

- **First term:** Observed feature count in training data
- **Second term:** Expected feature count under the model

At convergence, observed and expected feature counts match.

The expectation requires summing over all possible label sequences—computed efficiently using the **forward-backward algorithm** (analogous to HMM forward-backward).

Regularization. L2 regularization prevents overfitting:

$$\mathcal{L}_{\text{reg}}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$$

---

7 Relation Extraction

This section addresses extracting structured relations between entities.

7.1 Semantic Relations

Relation extraction identifies semantic relationships between named entities in text. Given a sentence with marked entities, the task is to determine what relation, if any, holds between entity pairs.

Example: From the sentence:

> *[PER Marie Curie] was born in [LOC Warsaw].*

Extract: born-in(Marie Curie, Warsaw)

Standard relation types from the ACE (Automatic Content Extraction) corpus include:

Relation	Description	Example
PHYS:Located	Physical location	*He was in Paris*
PHYS:Near	Physical proximity	*The store near the bank*
PER-SOC:Family	Family relationship	*John's mother*
ORG-AFF:Employment	Employment	*Google CEO*
PART-WHOLE	Part-whole relationship	*Door of the car*
GEN-AFF:Citizen	Citizenship	*French scientist*

7.2 Pattern-Based Extraction

The earliest relation extraction systems used **lexico-syntactic patterns**. Hearst (1992) identified patterns for extracting hyponym (IS-A) relations:

Hearst Patterns for Hyponymy:

Pattern	Example	Extracted Relation
NP_0 such as NP_1	*diseases such as measles*	hyponym(measles, disease)
such NP_0 as NP_1	*such authors as Hemingway*	hyponym(Hemingway, author)
NP_1 or other NP_0	*bruises or other injuries*	hyponym(bruises, injuries)
NP_1 and other NP_0	*Mozart and other composers*	hyponym(Mozart, composer)
NP_0, including NP_1	*European countries, including France*	hyponym(France, European country)
NP_0, especially NP_1	*herbs, especially basil*	hyponym(basil, herb)

Pattern Application. Given text containing:

> *Temple granaries, such as the that built by Rameses II, were used to store grain.*

Pattern matching extracts: hyponym(that built by Rameses II, Temple granaries)

Precision vs. Recall Trade-off. Patterns achieve high precision (extracted relations are usually correct) but low recall (many valid relations are not captured because they are expressed using different phrasings).

7.3 Supervised Relation Classification

Given labeled training data, relation extraction can be cast as **classification**: given two entities in a sentence, predict the relation type (including a NONE class).

Feature-Based Classification. Traditional approaches extract features from the entity pair and surrounding context:

Feature Category	Examples
Entity features	Entity types (PER, ORG), entity head words
Bag-of-words	Words between and around entities
Syntactic features	Dependency path between entities
Position features	Distance between entities

Neural Classification. Modern approaches use pretrained language models:

1. Mark entity positions with special tokens: [E1], [/E1], [E2], [/E2]
2. Encode: *[E1] Marie Curie [/E1] was born in [E2] Warsaw [/E2]*
3. Use [CLS] representation or pooled entity representations
4. Classify with softmax over relation types

7.4 Distant Supervision

Labeled relation data is expensive to create. **Distant supervision** (Mintz et al., 2009) generates training data automatically using knowledge bases.

Key Assumption: If two entities participate in a known relation in a knowledge base (e.g., Wikidata), then any sentence mentioning both entities likely expresses that relation.

Algorithm:

1. **Extract known relations** from knowledge base:

- born-in(Marie Curie, Warsaw)
- founded(Bill Gates, Microsoft)

2. **Find co-occurring entity pairs** in text corpus:
   - *Marie Curie was born in Warsaw* → born-in candidate
   - *Marie Curie died in Paris* → born-in false positive
3. **Train classifier** on automatically labeled data

Noise Reduction. The distant supervision assumption introduces noise—not every co-occurrence expresses the relation. Solutions include:

- Multi-instance learning (at least one sentence expresses the relation)
- Attention mechanisms to weight sentence contributions
- Explicit noise modeling

7.5 Bootstrapping

Bootstrapping (Brin, 1998; Agichtein & Gravano, 2000) iteratively learns patterns from seeds:

1. **Initialize** with seed entity pairs: {(Isaac Newton, physicist), (Albert Einstein, physicist)}
2. **Find sentences** containing seed pairs
3. **Extract patterns** from these sentences
4. **Apply patterns** to find new entity pairs
5. **Repeat** from step 2

Pattern Confidence. Patterns are scored by their precision on known examples:

$$\text{Conf}_{\text{RlogF}}(p) = \frac{|\text{hits}(p)|}{|\text{finds}(p)|} \cdot \log(|\text{finds}(p)|)$$

where:

- $|\text{hits}(p)|$ = number of pattern matches that are known correct pairs
- $|\text{finds}(p)|$ = total number of pattern matches

Tuple Confidence. Entity pair confidence combines evidence from multiple patterns using noisy-or:

$$\text{Conf}(t) = 1 - \prod_{p \in P}(1 - \text{Conf}(p))$$

This formula assumes patterns provide independent evidence.

7.6 Open Information Extraction

Open Information Extraction (Open IE) extracts relations without predefined relation types. Instead of classifying into fixed categories, it extracts arbitrary (subject, relation, object) triples.

ReVerb (Fader et al., 2011) uses syntactic patterns:

- Relations are verb phrases: *verb* | *verb prep* | *verb noun prep*
- Example: *Einstein developed the theory of relativity* → (Einstein, developed, the theory of relativity)

Constraints:

1. Relation phrase must begin with verb
2. Relation phrase must not be too long
3. Relation phrase should satisfy syntactic constraints

OLLIE extends ReVerb to handle nominal and implicit relations:

- *Bell, a telecommunications company* → (Bell, is, a telecommunications company)
- *The Curie-born scientist* → (scientist, born in, Curie)

---

8 Event and Temporal Extraction

This section addresses extraction of events and temporal information.

8.1 Event Detection

An **event** is a specific occurrence involving participants—something that happens at a particular time and place. Event extraction identifies:

- **Event trigger:** The word or phrase indicating the event
- **Event arguments:** Participants and their roles
- **Event type:** Classification of the event

Example: From ACE event annotation:

> *Last week [TIME], the company [ORG] announced [TRIGGER] the merger [THEME].*

Event: BUSINESS:DECLARE-BANKRUPTCY

- Trigger: *announced*
- Time: *Last week*
- Organization: *the company*

Event Types. Standard ontologies define event categories:

Type	Subtypes	Example Triggers
LIFE	Birth, Death, Marry	*born*, *died*, *married*
MOVEMENT	Transport	*went*, *arrived*, *flew*
TRANSACTION	Transfer-Money	*bought*, *paid*, *donated*
BUSINESS	Start-Org, Merge-Org	*founded*, *merged*
CONFLICT	Attack, Demonstrate	*attacked*, *protested*
JUSTICE	Arrest, Trial	*arrested*, *convicted*

8.2 Temporal Expressions

Temporal expressions denote points or intervals in time. The task involves:

1. **Detection:** Identifying temporal expression spans
2. **Normalization:** Mapping to canonical representations

TimeML/TIMEX3 is the standard annotation scheme:

Expression Type	Example	ISO Normalization
Fully specified	*January 15, 2024*	2024-01-15
Relative	*last Friday*	(computed from document time)
Duration	*three weeks*	P3W
Set	*every Monday*	XXXX-WXX-1

Relative Expressions. Many temporal expressions require anchoring to a reference time (typically document creation date):

- *yesterday* → document date - 1 day
- *next month* → document month + 1
- *last summer* → previous year's June-August

8.3 Temporal Normalization

Temporal normalization converts expressions to **ISO 8601** format:

Component	Format	Example
Date	YYYY-MM-DD	2024-01-15
Time	THH:MM:SS	T14:30:00
Duration	P[n]Y[n]M[n]DT[n]H[n]M[n]S	P2Y3M (2 years, 3 months)
Week	YYYY-Www	2024-W03 (week 3)
Quarter	YYYY-Qq	2024-Q1 (first quarter)

Underspecification. Some expressions are intentionally vague:

- *the 1990s* → 199X
- *last century* → 19XX
- *the evening* → TXXX (time unspecified)

8.4 Allen's Interval Algebra

Allen's interval algebra (Allen, 1983) provides a formal framework for temporal reasoning. Given two time intervals, exactly one of 13 mutually exclusive relations holds:

The 13 Allen Relations:

Relation	Symbol	Inverse	Definition
X **before** Y	<	> (after)	X ends before Y begins
X **meets** Y	m	mi	X ends exactly when Y begins
X **overlaps** Y	o	oi	X starts before Y, they overlap, X ends during Y
X **starts** Y	s	si	X and Y start together, X ends first
X **during** Y	d	di (contains)	X is contained within Y

Relation	Symbol	Inverse	Definition
X **finishes** Y	f	fi	X starts during Y, they end together
X **equals** Y	=	=	X and Y have same start and end

Visual Representation:

```
Before:      X: [----]
             Y:         [----]

Meets:       X: [----]
             Y:      [----]

Overlaps:    X: [----]
             Y:    [----]

Starts:      X: [--]
             Y: [------]

During:      X:   [--]
             Y: [------]

Finishes:    X:     [--]
             Y: [------]

Equals:      X: [----]
             Y: [----]
```

Temporal Reasoning. Allen relations support inference. If A *before* B and B *before* C, then A *before* C. Constraint propagation enables reasoning about complex temporal scenarios.

TempEval. The TempEval shared task evaluates:

1. Classifying temporal expressions relative to events
2. Identifying event-document time relations
3. Ordering events in a timeline

8.5 TimeBank

TimeBank is the primary corpus for temporal annotation, containing:

- Over 7,000 temporal expressions

- Event annotations with TimeML tags
- Temporal relation annotations (TLINK)

TLINK Relations encode temporal ordering:

- BEFORE, AFTER
- INCLUDES, IS_INCLUDED
- SIMULTANEOUS
- IDENTITY (same event)

---

9 Template Filling

This section addresses structured information extraction into predefined schemas.

Template filling extracts information to populate slots in predefined templates. Unlike relation extraction (binary relations) or event extraction (event-centric), template filling is schema-driven.

Example Template (Management Succession):

Slot	Filler
ORGANIZATION	Apple Inc.
POST	CEO
PERSON-IN	Tim Cook
PERSON-OUT	Steve Jobs
REASON-IN	succession
REASON-OUT	resignation

Template Slots. Each template type defines required and optional slots:

- **Required:** Must be filled for valid extraction
- **Optional:** May be absent
- **Set-valued:** Can have multiple fillers

Extraction Approaches:

1. **Rule-based (FASTUS architecture):**
    - Tokenization → Named entity recognition → Parsing → Merging → Template filling
2. **Machine learning:**
    - Train classifiers for slot filling
    - Encode candidates with contextual representations
    - Classify slot-filler pairs
3. **Neural reading comprehension:**
    - Frame slot filling as question answering
    - "Who became CEO?" → *Tim Cook*
    - "Which organization?" → *Apple Inc.*

---

10 Conclusion

This section summarizes the key concepts and their interconnections.

This chapter has developed the theory and methods for sequence labeling and information extraction in NLP. The key contributions are:

1. **Sequence labeling** generalizes classification from document-level to token-level predictions. POS tagging and NER are canonical tasks, with BIO tagging enabling span identification through per-token labels.
2. **Hidden Markov Models** provide a generative framework with transition probabilities $P(t_i|t_{i-1})$ capturing tag dependencies and emission probabilities $P(w_i|t_i)$ modeling word generation. The Markov and output independence assumptions enable tractable inference.
3. **The Viterbi algorithm** solves HMM decoding in $O(TN^2)$ time via dynamic programming. The recurrence $v_t(j) = \max_i v_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$ computes optimal path probabilities, with backpointers enabling sequence recovery.
4. **Conditional Random Fields** overcome generative model limitations by directly modeling $P(Y|X)$. Arbitrary feature functions—word shapes, prefixes, neighboring words—can be

incorporated while maintaining tractable Viterbi inference.

5. **Relation extraction** connects entities via semantic relations. Pattern-based methods achieve high precision; supervised and distant supervision approaches improve recall; Open IE extracts unconstrained relations.

6. **Temporal analysis** reasons about events and their temporal ordering. Allen's interval algebra provides formal foundations with 13 mutually exclusive relations.

These methods form the foundation for practical NLP systems in named entity recognition, part-of-speech tagging, and information extraction pipelines.

---

11 References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 832-843.

Brin, S. (1998). Extracting patterns and relations from the World Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases*.

Fader, A., Soderland, S., & Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of EMNLP*.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*.

Jurafsky, D., & Martin, J. H. (2024). *Speech and Language Processing* (3rd ed.). Chapters 17 and 20.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), 260-269.

Appendix A: Viterbi Algorithm Pseudocode

```
function VITERBI(observations, states, A, B, π):
    # Initialization
    for each state s in states:
        viterbi[0, s] = π[s] * B[s, observations[0]]
        backpointer[0, s] = 0

    # Recursion
    for t = 1 to T-1:
        for each state s in states:
            viterbi[t, s] = max over s' of:
                viterbi[t-1, s'] * A[s', s] * B[s, observations[t]]
            backpointer[t, s] = argmax over s' of:
                viterbi[t-1, s'] * A[s', s] * B[s, observations[t]]

    # Termination
    best_path_prob = max over s of: viterbi[T-1, s]
    best_last_state = argmax over s of: viterbi[T-1, s]

    # Backtracking
    best_path = [best_last_state]
    for t = T-1 down to 1:
        best_path.prepend(backpointer[t, best_path[0]])

    return best_path, best_path_prob
```

Complexity Analysis:

- Time: $O(TN^2)$ — T time steps, N states per step, N transitions per state
- Space: $O(TN)$ — storing Viterbi probabilities and backpointers

Appendix B: CRF Feature Template Examples

Lexical Features:

```
# Word identity
feature: word[i] = X, label = Y
```

```
feature: word[i-1] = X, label = Y
feature: word[i+1] = X, label = Y

# Word combinations
feature: word[i-1] = X, word[i] = Z, label = Y
```

Orthographic Features:

```
# Capitalization
feature: isCapitalized(word[i]), label = Y
feature: isAllCaps(word[i]), label = Y
feature: hasMixedCase(word[i]), label = Y

# Morphology
feature: prefix(word[i], 3) = X, label = Y  # 3-char prefix
feature: suffix(word[i], 3) = X, label = Y  # 3-char suffix

# Word shape
feature: shape(word[i]) = X, label = Y
```

Transition Features:

```
# Label bigrams
feature: label[i-1] = X, label[i] = Y

# Label + word
feature: label[i-1] = X, label[i] = Y, word[i] = Z
```

Glossary

Term	Definition
Allen's Interval Algebra	Formal framework with 13 mutually exclusive temporal relations between intervals
BIO Tagging	Labeling scheme: B (beginning), I (inside), O (outside) for span identification
Conditional Random Field (CRF)	Discriminative sequence model directly estimating $P(Y
Distant Supervision	Training data generation using knowledge base relations as noisy labels
Emission Probability	$P(w_i
Feature Function	Function extracting features from input-label pairs for CRF scoring

Term	Definition
Generative Model	Model of joint probability $P(X, Y)$ over inputs and labels
Hearst Patterns	Lexico-syntactic patterns for extracting hyponym relations
Hidden Markov Model (HMM)	Generative sequence model with hidden states and observed emissions
Markov Assumption	$P(q_i
Named Entity Recognition (NER)	Task of identifying and classifying named entities in text
Open Information Extraction	Relation extraction without predefined relation types
Output Independence	$P(o_i
Part-of-Speech (POS) Tagging	Task of assigning grammatical categories to words
Partition Function	$Z(X) = \sum_{Y'} \exp(\sum_k w_k F_k(X, Y'))$: normalization constant for CRF
Relation Extraction	Task of identifying semantic relations between entities
Sequence Labeling	Assigning a label to each element in a sequence
Template Filling	Populating predefined schema slots from text
Temporal Expression	Linguistic expression denoting time (point, interval, duration)
TimeML	Markup language for temporal annotation
Transition Probability	$P(t_i
Viterbi Algorithm	Dynamic programming algorithm for HMM/CRF decoding in $O(TN^2)$ time