IA312: Structured prediction in natural language processing

Maria Boritchev

March 18, 2024

Télécom Paris, Institut Polytechnique de Paris

Introduction

What is structured prediction?

Tasks: predict structured outputs $\neq$ scalar value/categorical quantity

What is structured prediction?

Tasks: predict structured outputs $\neq$ scalar value/categorical quantity

What NLP tasks do you know?

What is structured prediction?

Tasks: predict structured outputs $\neq$ scalar value/categorical quantity

What NLP tasks do you know?

Not structured prediction sentiment analysis, text classification, chatbots, autocorrection, speech recognition

Structured prediction tagging, parsing, coreference resolution, text summarization

Depends? machine translation, natural language understanding, natural language inference

Outline

Part-of-Speech (POS) tagging

Named Entity Recognition (NER)

Algorithms

Dependency parsing

Part-of-Speech (POS) tagging

Part-of-Speech (POS) tagging

Example:

Charlie likes green tea

Part-of-Speech (POS) tagging

Example:

Charlie	likes	green	tea
NNP	VBZ	JJ	NN

Part-of-Speech (POS) tagging

Example:

Charlie	likes	green	tea
NNP	VBZ	JJ	NN

What do the tags stand for?

Part-of-Speech (POS) tagging

Example:

Charlie	likes	green	tea
NNP	VBZ	JJ	NN

What do the tags stand for?

NNP	Proper noun
VBZ	Verb
JJ	Adjective
NN	Noun

Why is POS tagging interesting and useful?

Computational linguistics tasks:

- **Linguistic change:** creation of new words, meaning shift
- **Linguistic variation:** regional, social, medical
- **Comparison and control:** use POS to measure meaning similarity

Why is POS tagging interesting and useful?

Computational linguistics tasks:

- **Linguistic change:** creation of new words, meaning shift
- **Linguistic variation:** regional, social, medical
- **Comparison and control:** use POS to measure meaning similarity

NLP tasks:

- Syntactic parsing
- **Machine translation:** word order (SOV/SVO/VSO etc.)
- **Sentiment analysis:** adjectives vs. other POS
- **Text-to-speech:** pronunciation disambiguation

POS tags – Universal Dependencies (UD)

ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
INTJ	interjection
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other

Is POS tagging a difficult task in English?

Ambiguity: $\sim 15\%$ of words are ambiguous

Is POS tagging a difficult task in English?

Ambiguity: $\sim 15\%$ of words are ambiguous
Example: "back"

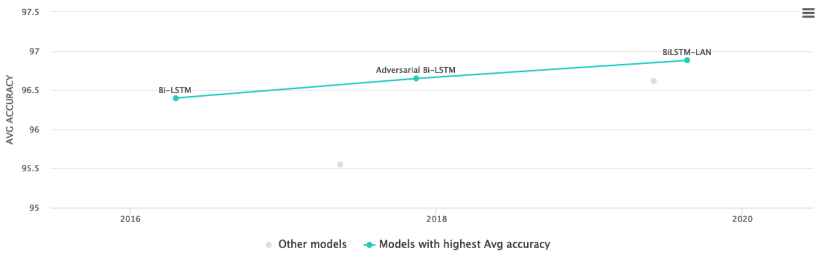
- Earnings growth took a back/**ADJ** seat
- A small building in the back/**NOUN**
- A clear majority of senators back/**VERB** the bill
- Enable the country to buy back/**PART** debt
- I was twenty-one back/**ADV** then

SOTA performance of POS tagging in English

Part-Of-Speech Tagging on UD

Leaderboard   Dataset

View: Avg accuracy by: Date



Filter: LSTM untagged

Edit Leaderboard

Rank	Model	Avg accuracy	Paper	Code	Result	Year	Tags
1	**BiLSTM-LAN**	96.88	Hierarchically-Refined Label Attention Network for Sequence Labeling			2019	LSTM

Garden path sentences

Example: “The horse raced past the barn fell.”
Exemple: « L’oiseau vola tout en douceur au bord de la fenêtre le diamant qui y était déposé. »

Garden path sentences

Example:

- The/**DET** horse/**NOUN** raced/**VERB** past/**NOUN** the/**DET** barn/**NOUN** fell/**VERB**
- **Explanation:** “the horse **which was** raced past the barn fell”

Example: « L’oiseau vola tout en douceur au bord de la fenêtre le diamant qui y était déposé »

Sources of information for POS tagging

Example: "Janet will back the bill"

Sources of information for POS tagging

Example: "Janet will back the bill"

- "will": AUX/NOUN/VERB?
- "bill": NOUN/VERB?

Sources of information for POS tagging

Example: "Janet will back the bill"

- "will": **AUX**/**NOUN**/**VERB**?
- "bill": **NOUN**/**VERB**?

Some contextual clues:

- Prior probabilities of word/tag:
  - "will" is usually an **AUX**
- Neighboring words:
  - "the" means the next word is probably not a **VERB**
- Morphology and wordshape:

Prefixes:	un- $\rightarrow$ **ADJ**	ex: unable
Suffixes:	-ly $\rightarrow$ **ADJ**	ex: importantly
Capitalization:	CAP $\rightarrow$ **PROPN**	ex: Janet

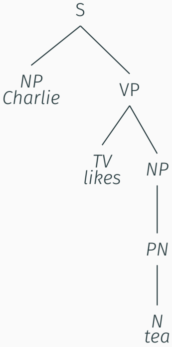
Exercise/example: simplified POS tags

Abbreviation	Name	Example
N	noun	“tea”
PN	proper noun	“Charlie”
TV	transitive verb	“likes”
Adj	adjective	“green”
D	determiner	“the”

Abbreviation	Name	Example
NP	noun phrase	“the unicorn”
VP	verb phrase	“likes tea”
PP	prepositional phrase	“with a teapot”
S	sentence	“Charlie likes tea”

Exercise/example: “Charlie likes tea” – A minimal grammar

$$\text{Lexicon:} \begin{cases} PN & \rightarrow & Charlie \\ N & \rightarrow & tea \\ TV & \rightarrow & likes \end{cases} \qquad \text{Grammar:} \begin{cases} S & \rightarrow & NP\ VP \\ VP & \rightarrow & TV\ NP \\ NP & \rightarrow & PN \\ PN & \rightarrow & N \end{cases}$$



Algorithms for POS tagging

Supervised Machine Learning Algorithms:

- Hidden Markov Models (HMM)
- Conditional Random Fields (CRF)
- Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Finetuned LLMs (ex: BERT)

⚠ All of these required a human-labeled training set!

Named Entity Recognition (NER)

What are Named Entities?

A **named entity** (NE) is anything that can be referred to with a proper name. Most common NE are of one of the four types:

- **People:** “Marie Curie”
- **Locations:** “Paris”
- **Organizations:** “Institut Polytechnique de Paris”
- **Geo-Political Entities:** “European Union”

The term is now extended: multi-word expressions, dates, times, prices.

Exercise/example: NE tagging/NER

Task:

(1) find spans of text constituting the NE;

(2) tag the type (PEO, LOC, ORG, GPE, AMOUNT) of this NE.

Example:

The STA Plaza (The Plaza or Spokane Transit Authority Plaza), is a transit center located in Downtown Spokane, Washington. It is the main hub of customer service and transit operations for the Spokane Transit Authority (STA), with 31 out of its 52 bus routes connecting with The Plaza.

Exercise/example: NE tagging/NER

Task:

(1) find spans of text constituting the NE;

(2) tag the type (PEO, LOC, ORG, GPE, AMOUNT) of this NE.

(1):

The [STA Plaza] ([The Plaza] or [Spokane Transit Authority Plaza]), is a transit center located in [Downtown Spokane, Washington]. It is the main hub of customer service and transit operations for the [Spokane Transit Authority (STA)], with [31] out of its [52] bus routes connecting with [The Plaza].

Exercise/example: NE tagging/NER

Task:

(1) find spans of text constituting the NE;

(2) tag the type (PEO, LOC, ORG, GPE, AMOUNT) of this NE.

(2):

The STA Plaza/LOC (The Plaza/LOC or Spokane Transit Authority Plaza/LOC), is a transit center located in Downtown Spokane, Washington/LOC. It is the main hub of customer service and transit operations for the Spokane Transit Authority (STA)/ORG, with 31/AMOUNT out of its 52/AMOUNT bus routes connecting with The Plaza/LOC.

Why is NER interesting and useful?

Examples in NLP applications:

- **Sentiment analysis:** consumer's sentiment towards a particular company or person?
- **Question answering:** answer questions about an entity?
- **Information extraction:** extracting facts about entities from text

Is NER a difficult task in English?

Segmentation: where does a NE start/end?

Type ambiguity: can a NE be of different types depending on the context?

Examples:

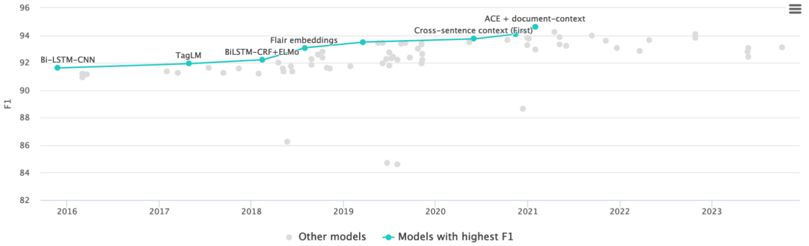
- *Washington/PER was born into slavery on the farm of James Burroughs.*
- *Washington/ORG went up 2 games to 1 in the four-game series.*
- *Blair arrived in Washington/LOC for what may well be his last state visit.*

SOTA performance of NER in English

Named Entity Recognition (NER) on CoNLL 2003 (English)

Leaderboard   Dataset

View F1 by Date for All models



Filter: LSTM Transformer MRC-based knowledge distillation GCN untagged

Edit Leaderboard

Rank	Model	F1	Extra Training Data	Paper	Code	Result	Year	Tags
1	**ACE + document-context**	94.6	×	Automated Concatenation of Embeddings for Structured Prediction			2021	LSTM Transformer

An algorithm for NER

Beginning-inside-outside (BIO) tagging: merging (1) and (2) NER tasks into one tagging task.

Example:

Charlie	is	going	to	Los	Angeles

An algorithm for NER

Beginning-inside-outside (BIO) tagging: merging (1) and (2) NER tasks into one tagging task.

Example:

Charlie	is	going	to	Los	Angeles
I-PER	O	O	O	B-LOC	I-LOC

Algorithms for NER

Supervised Machine Learning Algorithms:

- Hidden Markov Models (HMM)
- Conditional Random Fields (CRF)
- Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Finetuned LLMs (ex: BERT)

⚠ All of these required a human-labeled training set!

Coreference resolution

Coreference resolution (CR) is the task of finding all linguistic expressions in a given text that refer to the same entity.

Example:
Sam saw a white star twinkle for a while. The beauty of it smote his heart, as he looked up out of the forsaken land, and hope returned to him.

Coreference resolution

Coreference resolution (CR) is the task of finding all linguistic expressions in a given text that refer to the same entity.

Example:
Sam saw a white star twinkle for a while. The beauty of it smote his heart, as he looked up out of the forsaken land, and hope returned to him.

Algorithms

A **sequence labeler** is a model that assigns a label to each unit in a sequence, mapping a sequence of observations to a sequence of labels of the same length.

Sequence labeling algorithms

A **sequence labeler** is a model that assigns a label to each unit in a sequence, mapping a sequence of observations to a sequence of labels of the same length.

Examples:

Charlie	likes	green	tea
NNP	**VBZ**	**JJ**	**NN**

Charlie	is	going	to	Los	Angeles
I-PER	**O**	**O**	**O**	**B-LOC**	**I-LOC**

Hidden Markov Model (HMM)

Probabilistic graphical model: given a sequence of units (words, letters, morphemes, sentences, etc.), it computes a probability distribution over possible sequences of labels and assigns the best label sequence.

Markov chain

A **Markov chain** is a model based on the assumption that the next state in a sequence only depends on the current state.

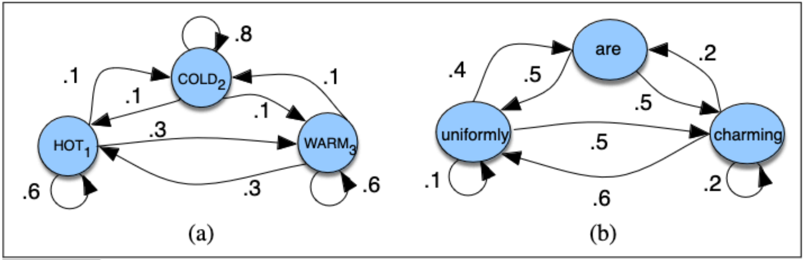


Figure 8.8 A Markov chain for weather (a) and one for words (b), showing states and transitions. A start distribution π is required; setting $\pi = [0.1, 0.7, 0.2]$ for (a) would mean a probability 0.7 of starting in state 2 (cold), probability 0.1 of starting in state 1 (hot), etc.

Markov chain

A **Markov chain** is a model based on the assumption that the next state in a sequence only depends on the current state.

Markov Assumption: $P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_{i-1})$

Markov chain

Specification of a Markov chain:

$Q = q_1 q_2 \dots q_N$	a set of N **states**
$A = a_{11} a_{12} \dots a_{N1} \dots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have π_j = 0, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1 \forall i$

Hidden Markov Model

A **Hidden Markov Model** allows us to compute a probability of both **observed events** (such as words that we see in the input) and **hidden events** (such as part-of-speech tags) that we think of as causal factors in our probabilistic model.

Hidden Markov model

Specification of an HMM:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1 v_2 \ldots v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have π_j = 0, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1 \forall i$

HMM: two assumptions

The probability of a particular state depends only on the previous state:

$$\textbf{Markov Assumption: } P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_{i-1})$$

The probability of an output observation o_i depends only on the state that produced the observation q_i and not on any other states or any other observations:

$$\textbf{Output Independence: } P(o_i | q_i, \ldots, q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i | q_i)$$

HMM tagger

An HMM has two components:

$$\lambda = (A, B)$$

A is a matrix that contains the tag **transition probabilities** $P(t_i|t_{i1})$: the probability of a tag occurring given the previous tag.

B is a matrix that contains the **emission probabilities** or **observation likelihoods** $P(w_i|t_i)$: the probability, given a tag, that it will be associated with a given word

HMM tagger: $\lambda = (A, B)$

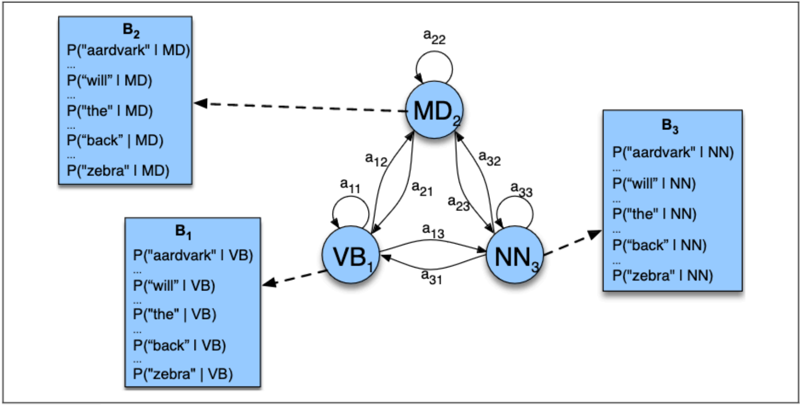


Figure 8.9 An illustration of the two parts of an HMM representation: the A transition probabilities used to compute the prior probability, and the B observation likelihoods that are associated with each state, one likelihood for each possible observation word.

HMM tagging as decoding

Decoding: determining the hidden variables sequence corresponding to the sequence of observations.

$\rightarrow$ Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \ldots, o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$.

HMM decoding for POS tagging

Goal: choose the tag sequence $t_1 \dots t_n$ that is most probable given the observation sequence of n words $w_1 \dots w_n$:

$$\hat{t}_{1:n} = argmax_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n)$$

The decoding algorithm for HMMs is the **images/Viterbi algorithm**.

Viterbi algorithm

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow \pi_s \ * \ b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} \ viterbi[s',t-1] \ * \ a_{s',s} \ * \ b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \ viterbi[s',t-1] \ * \ a_{s',s} \ * \ b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} \ viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \ viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure 8.10 Viterbi algorithm for finding the optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

Visualisations of Viterbi algorithm

NLP context:
https://www.youtube.com/watch?v=Lj4NKxg0xa0

Shortest path context:
https://www.youtube.com/watch?v=6JVqutwtzmo

Lattice/Probability matrix `viterbi[N,T]`

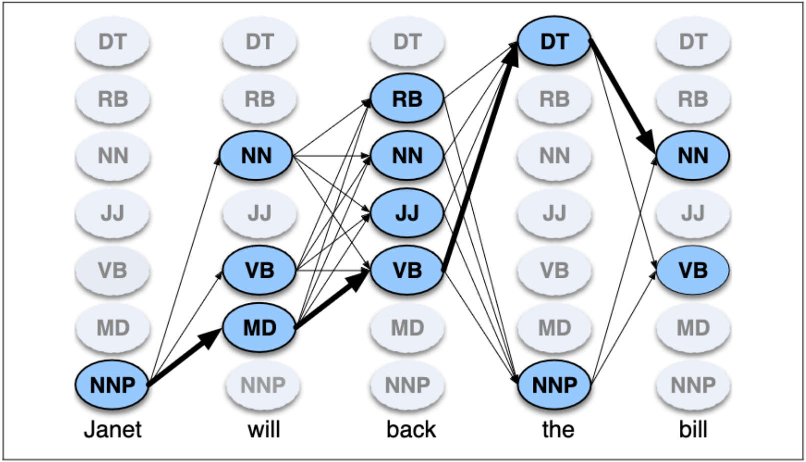


Figure 8.11 A sketch of the lattice for *Janet will back the bill*, showing the possible tags (q_i) for each word and highlighting the path corresponding to the correct tag sequence through the hidden states. States (parts of speech) which have a zero probability of generating a particular word according to the *B* matrix (such as the probability that a determiner DT will be realized as *Janet*) are greyed out.

CRF and MEMM

Goal: add information about the context in which a word appears.

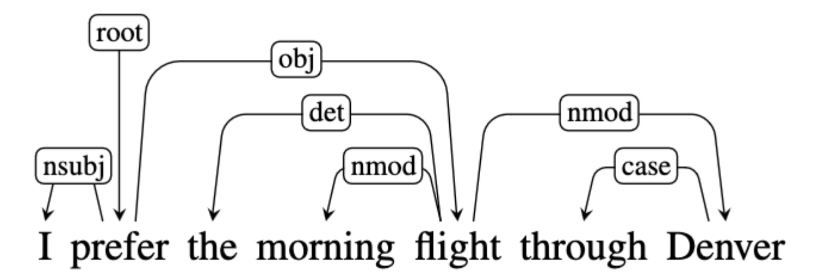
- → What is the previous word? **AND** the next word?
- → What is the syntactic shape of the sentence parts before and after the word?
- → Are there unknown words?

Conditional Random Fields (CRF) and **Maximum Entropy Markov Models** (MEMM) allow integration of rich features for better accuracy.
Cost: much slower training!

Dependency parsing

Dependency parsing

In **dependency parsing**, the syntactic structure of a sentence is described in terms of directed binary grammatical relations between the words:



The arcs go from **heads** to **dependents**; the parse is a **typed dependency structure**.

→ Dependency structure shows which words depend on (modify, attach to, or are arguments of) which other words.

Why is dependency parsing interesting and useful?

Computational linguistics tasks:

- **Human communication:** compositionality for complex sentences and meanings
- **Linguistic variation:** multilingual, regional, social, medical

Why is dependency parsing interesting and useful?

Computational linguistics tasks:

- **Human communication:** compositionality for complex sentences and meanings
- **Linguistic variation:** multilingual, regional, social, medical

NLP tasks:

- Syntactic parsing for **semantic parsing**
- **Machine translation:** word order (SOV/SVO/VSO etc.)
- **Sentiment analysis:** adjectives vs. other POS
- **Text-to-speech:** pronunciation disambiguation

Phrase attachment ambiguities

Examples:

- Scientists count whales from space
- Teacher strikes idle kids
- Enraged cow injures farmer with axe
- Miners refuse to work after death

https://www.departments.bucknell.edu/linguistics/synhead.html

Dependency formalism

A **dependency structure** can be represented as a directed graph $G = (V, A)$:

- a set of vertices V ($\sim$ set of words in a given sentence + punctuation)
- a set of ordered pairs of vertices A (arcs)

Dependency formalism

A **dependency structure** can be represented as a directed graph $G = (V, A)$:

- a set of vertices V ($\sim$ set of words in a given sentence + punctuation)
- a set of ordered pairs of vertices A (arcs)

Different grammatical theories or formalisms may place further constraints on these dependency structures:

- the structures must be connected
- have a designated root node
- be acyclic or planar.

A **dependency tree** is a directed graph that satisfies the following constraints:

1. There is a single designated root node that has no incoming arcs.
2. With the exception of the root node, each vertex has exactly one incoming arc.
3. There is a unique path from the root node to each vertex in V.

Thus:

- $\rightarrow$ each word has a single head;
- $\rightarrow$ the dependency structure is connected;
- $\rightarrow$ there is a single root node from which one can follow a unique directed path to each of the words in the sentence.

Dependency relations

Clausal Argument Relations	Description
NSUBJ	Nominal subject
OBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Figure 18.2 Some of the Universal Dependency relations (de Marneffe et al., 2021).

Dependency relations examples

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Figure 18.3 Examples of some Universal Dependency relations.

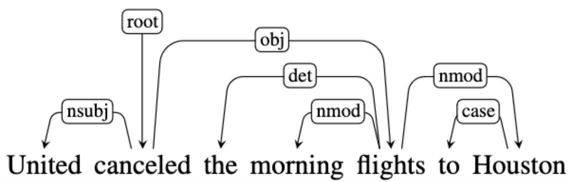
Exercise/example

Relation	Description & example (with head and dependent)
ROOT	Root of the tree, head of the entire structure
NSUBJ	Nominal subject, ex: **United** canceled the flight.
OBJ	Direct object, ex: We booked her the first **flight** to Miami.
NMOD	Nominal modifier, ex: We took the **morning** flight.
DET	Determiner, ex: **The** flight was canceled.
CASE	Prepositions, postpositions, other case markers, ex: Book the flight **through** Houston.

Exercise: What is the dependency parse of the sentence "United canceled the morning flights to Houston"?

Exercise/example

Relation	Description & example (with head and dependent)
ROOT	Root of the tree, head of the entire structure
NSUBJ	Nominal subject, ex: **United** canceled the flight.
OBJ	Direct object, ex: We booked her the first **flight** to Miami.
NMOD	Nominal modifier, ex: We took the **morning** flight.
DET	Determiner, ex: **The** flight was canceled.
CASE	Prepositions, postpositions, other case markers, ex: Book the flight **through** Houston.



Universal Dependencies project - more than 100 languages

Current UD Languages

Information about language families (and genera for families with multiple branches) is mostly taken from WALS Online (IE = Indo-European).

Abaza	1	<1K	Northwest Caucasian
Afrikaans	1	49K	IE, Germanic
Akkadian	2	25K	Afro-Asiatic, Semitic
Akuntsu	1	1K	Tupian, Tupari
Albanian	1	<1K	IE, Albanian
Amharic	1	10K	Afro-Asiatic, Semitic
Ancient Greek	2	416K	IE, Greek
Ancient Hebrew	1	39K	Afro-Asiatic, Semitic
Apurina	1	<1K	Arawakan
Arabic	3	1,042K	Afro-Asiatic, Semitic
Armenian	2	94K	IE, Armenian
Assyrian	1	<1K	Afro-Asiatic, Semitic
Bambara	1	13K	Mande
Basque	1	121K	Basque
Beja	1	<1K	Afro-Asiatic, Cushitic
Belarusian	1	305K	IE, Slavic
Bengali	1	<1K	IE, Indic
Bhojpuri	1	6K	IE, Indic
Bororo	1	<1K	Bororoan
Breton	1	10K	IE, Celtic
Bulgarian	1	156K	IE, Slavic
Buryat	1	10K	Mongolic
Cantonese	1	13K	Sino-Tibetan
Catalan	1	553K	IE, Romance
Cebuano	1	1K	Austronesian, Central Philippine
Chinese	6	287K	Sino-Tibetan
Chukchi	1	6K	Chukotko-Kamchatkan
Classical Chinese	1	433K	Sino-Tibetan
Coptic	1	55K	Afro-Asiatic, Egyptian

Dependency treebanks

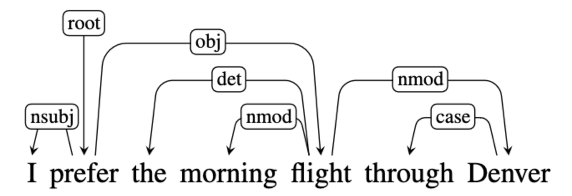
Treebanks play a critical role in the development and evaluation of dependency parsers:

- used for training parsers;
- act as the gold labels for evaluating parsers;
- provide useful information for corpus linguistics studies.

Dependency treebanks are created by having human annotators directly create dependency structures for a given corpus, or by hand-correcting the output of an automatic parser.

Sources of information for dependency parsing

Example:



- **Bilexical affinities:** the dependency [flight → morning] is plausible
- **Dependency distance:** Most dependencies are between nearby words
- **Intervening material:** Dependencies rarely span intervening words (ex: like, with, plus, including) or punctuation
- **Valency of heads:** How many dependents on which side are usual for a head?

Transition-based dependency parsing

Three components:

(1) a **stack**, on which the parse is built

(2) a **buffer**, containing the tokens to be parsed

(3) a **parser** which takes actions on the parse via a predictor: an oracle

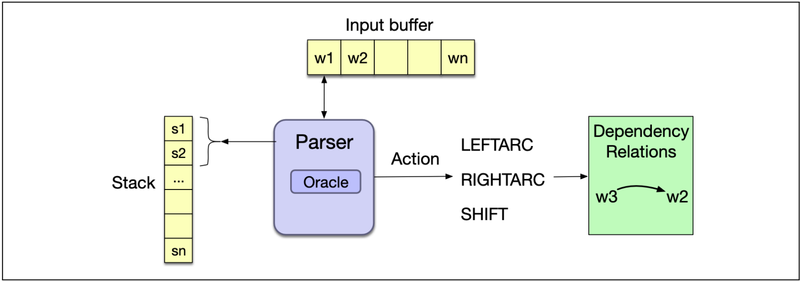


Figure 18.4 Basic transition-based parser. The parser examines the top two elements of the stack and selects an action by consulting an oracle that examines the current configuration.

Transition-based dependency parsing

The parser: walks through the sentence left-to-right, shifting items from the **buffer** onto the **stack**.

Transition-based dependency parsing

The parser: walks through the sentence left-to-right, shifting items from the **buffer** onto the **stack**.

At each time point: the top two elements on the **stack** are examined, the oracle makes a decision about what transition to apply to build the parse:

- *LEFTARC*: assign the current word as the head of some previously seen word
- *RIGHTARC*: assign some previously seen word as the head of the current word;
- *SHIFT*: postpone dealing with the current word, storing it for later processing.

Transition-based dependency parsing

The parser: walks through the sentence left-to-right, shifting items from the **buffer** onto the **stack**.

At each time point: the top two elements on the **stack** are examined, the oracle makes a decision about what transition to apply to build the parse:

- *LEFTARC*: assign the current word as the head of some previously seen word
- *RIGHTARC*: assign some previously seen word as the head of the current word;
- *SHIFT*: postpone dealing with the current word, storing it for later processing.

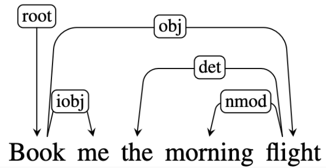
⚠ *LEFTARC* cannot be applied when ROOT is the second element of the stack.

⚠ *LEFTARC* and *RIGHTARC* require two elements to be on the stack to be applied.

Example: transition-based parse trace

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Figure 18.6 Trace of a transition-based parse.



Oracle

The **oracle** for greedily selecting the appropriate transition is trained by **supervised machine learning**: **need training data!**

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Figure 18.7 Generating training items consisting of configuration/predicted action pairs by simulating a parse with a given reference parse.

Evaluation

How to evaluate parsing?

Evaluation

How to evaluate parsing?

- **Exact match (EM):** how many sentences are parsed correctly
- $\rightarrow$ most sentences are marked wrong

Evaluation

How to evaluate parsing?

- **Exact match (EM):** how many sentences are parsed correctly
- $\rightarrow$ most sentences are marked wrong
- **Labeled attachment score (LAS):** is a word properly assigned to its head with the correct dependency relation?
- **Unlabeled attachment score (UAS):** is a word properly assigned to its head? (ignoring the dependency relation)
- **Label accuracy score (LS):** what is the percentage of tokens with correct labels? (ignoring where the relations come from)

How to evaluate parsing?

- **Exact match (EM):** how many sentences are parsed correctly
- $\rightarrow$ most sentences are marked wrong
- **Labeled attachment score (LAS):** is a word properly assigned to its head with the correct dependency relation?
- **Unlabeled attachment score (UAS):** is a word properly assigned to its head? (ignoring the dependency relation)
- **Label accuracy score (LS):** what is the percentage of tokens with correct labels? (ignoring where the relations come from)

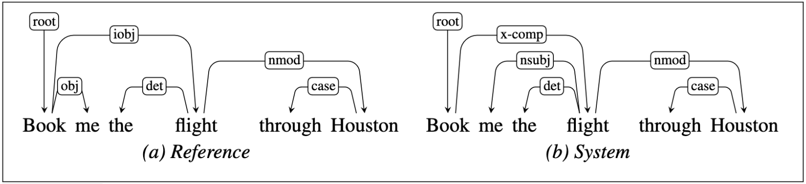


Figure 18.15 Reference and system parses for *Book me the flight through Houston*, resulting in an LAS of 2/3 and an UAS of 5/6.

Conclusion

To sum up

Structured prediction: prediction of structured outputs, such as

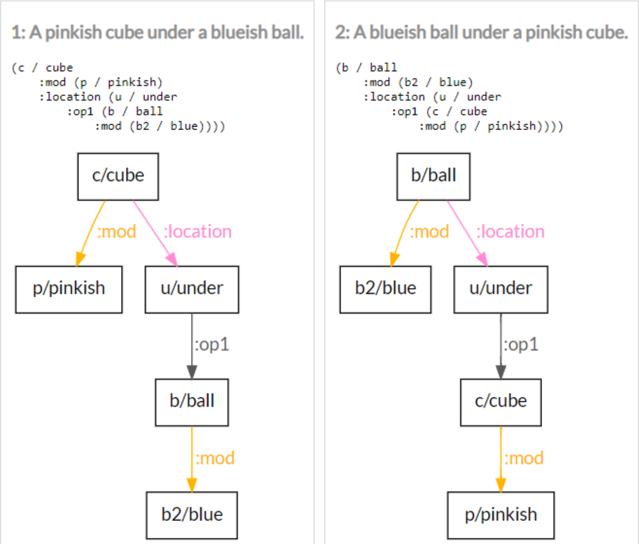
- **Part-of-speech** (POS) **tagging**
- **Named Entity Recognition** (NER)
- **Coreference resolution**

Historically, human-crafted rule-based grammars were used. Now, prediction algorithms such as **Hidden Markov Models (HMM)**, and more advanced ones such as Conditional Random Fields (CRF) and Maximum Entropy Markov Models (MEMM).

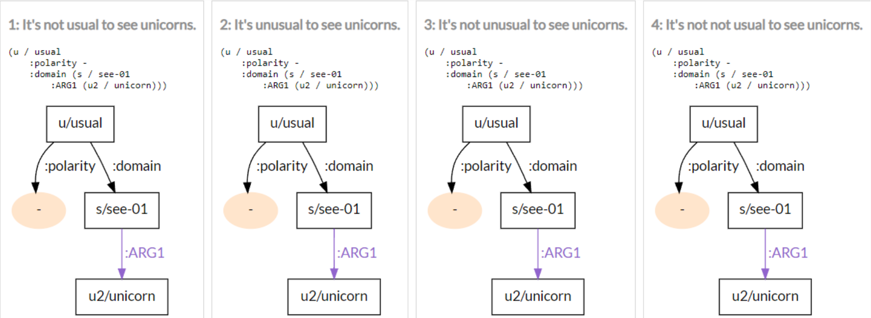
A more advanced example of a structured prediction task is **dependency parsing**, for which both feature-based models and prediction algorithms are needed.

Semantic parsing

AMR parsing [3, 1]:



Semantic parsing and semantics



To go further

Structured Prediction in NLP - A survey

Chauhan Dev Naman Biyani Nirmal P. Suthar
Prashant Kumar Priyanshu Agarwal
{devgiri, namanb, nirmalps, praskr, priyanag}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Over the last several years, the field of Structured prediction in NLP has had seen huge advancements with sophisticated probabilistic graphical models, energy-based networks, and its combination with deep learning-based approaches. This survey provides a brief of major techniques in structured prediction and its applications in the NLP domains like parsing, sequence labeling, text generation, and sequence to sequence tasks. We also deep-dived into energy-based and attention-based techniques in structured prediction, identified some relevant open issues and gaps in the current state-of-the-art research, and have come up with some detailed ideas for future research in these fields.

Nowadays

ACL 2022 6th Workshop on Structured Prediction for NLP

Call for Papers   Important Dates   Schedule   Accepted Papers   Organizers   Anti-Harassment Policy

Accepted Papers

Accepted Papers

- *A Joint Learning Approach for Semi-supervised Neural Topic Modeling.* Jeffrey Chiu, Rajat Mittal, Neehal Tumma, Abhishek Sharma and Finale Doshi-Velez
- *SlotGAN: Detecting Mentions in Text via Adversarial Distant Learning.* Daniel Daza, Michael Cochez and Paul Groth
- *TempCaps: A Capsule Network-based Embedding Model for Temporal Knowledge Graph Completion.* Guirong Fu, Zhao Meng, Zhen Han, Zifeng Ding, Yunpu Ma, Matthias Schubert, Volker Tresp and Roger Wattenhofer
- *Joint Entity and Relation Extraction Based on Table Labeling Using Convolutional Neural Networks.* Youmi Ma, Tatsuya Hiraoka and Naoaki Okazaki
- *Multilingual Syntax-aware Language Modeling through Dependency Tree Conversion.* Shunsuke Kando, Hiroshi Noji and Yusuke Miyao
- *Predicting Attention Sparsity in Transformers.* Marcos Vinicius Treviso, António Góis, Patrick Fernandes, Erick Rocha Fonseca, and Andre Martins
- *Neural String Edit Distance.* Jindřich Libovický and Alexander Fraser

Bibliography i

https://www.cs.cornell.edu/courses/cs6741/2015fa/

https://web.stanford.edu/~jurafsky/slp3/slides/8_POSNER_intro_May_6_2021.pdf

https://universaldependencies.org/u/pos/

https://paperswithcode.com/sota/part-of-speech-tagging-on-ud

https://paperswithcode.com/task/named-entity-recognition-ner

https://web.stanford.edu/~jurafsky/slp3/8.pdf

https://web.stanford.edu/~jurafsky/slp3/18.pdf

https://universaldependencies.org/

Bibliography ii

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture04-dep-parsing.pdf

https://en.wikipedia.org/wiki/K-means_clustering

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider.
Abstract meaning representation for sembanking.
In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186, 2013.

C. Dev, N. Biyani, N. P. Suthar, P. Kumar, and P. Agarwal.
Structured Prediction in NLP – A survey.

J. Heinecke and A. Shimorina.
Multilingual abstract meaning representation for celtic languages.
In *Proceedings of the 4th Celtic Language Technology Workshop within LREC2022*, pages 1–6, 2022.

D. Jurafsky and J. H. Martin.
Speech and language processing (3rd (draft) ed.), 2019.