Natural Language Processing

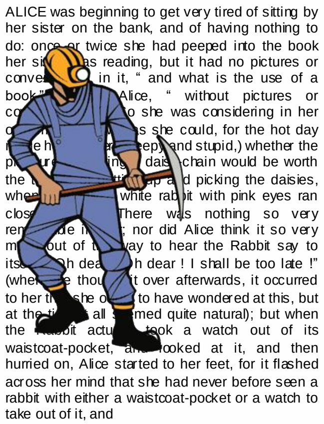
POLITECNICO DI MILANO

Sequence Classification and Sequence Labeling

Natural Language Processing

Some slide content based on textbooks:
***Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition** by Daniel Jurafsky and James H. Martin*

Mark Carman



Miner Image source: https://freesvg.org/miner-157-442488-4

Lecture Contents:

- Importance of Word Order
- Sequence Classification vs Labelling
- Traditional Sequence Labelling Models: HMMs and CRFs
- Recurrent Neural Networks
- NLP Applications:
  - POS tagging
  - Named Entity Recognition
  - Entity Linking
  - Relation Extraction
  - Parse Trees
  - Co-reference Resolution
  - Ontologies

Word order

Importance of word order

Word order super important for interpreting meaning of text

- and **for classifying it**

For example, consider the **intended meaning** of the following phrases:

- There's a **white** rat in the house …
- There's a rat in the **White** House …

Source: https://commons.wikimedia.org/wiki/File:White_House_DC.JPG

Negation provides a particularly important example of word order:

Which sentences is the overall sentiment positive?

- I am **not** happy about going to school tomorrow.
- I am happy about **not** going to school tomorrow.

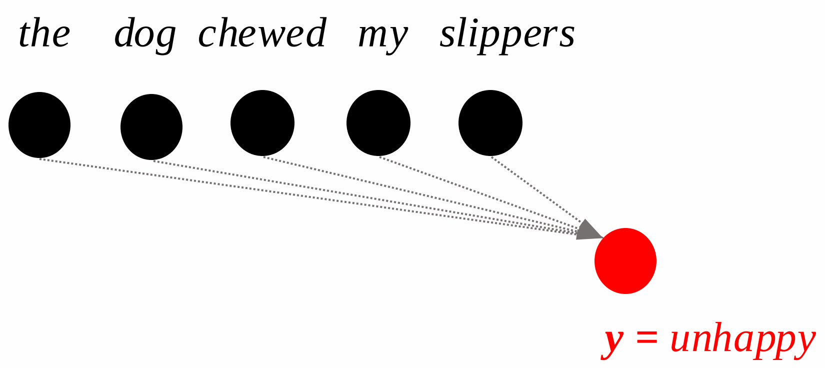
N-grams can indeed be used to capture word order information

- but we can never quite make them long enough …

Sequence Classification vs Sequence Labelling?

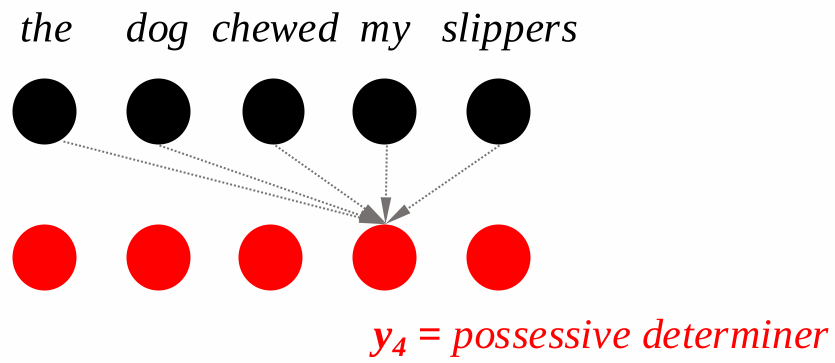
Sequence classification task:

- input: **ordered sequence** of tokens: $(w_1, w_2, \ldots, w_n)$
- output: single prediction for sequence: y



Sequence labelling task:

- input: **ordered sequence** of tokens: $(w_1, w_2, \ldots, w_n)$
- output **sequence of predictions**: $(y_1, y_2, \ldots, y_n)$

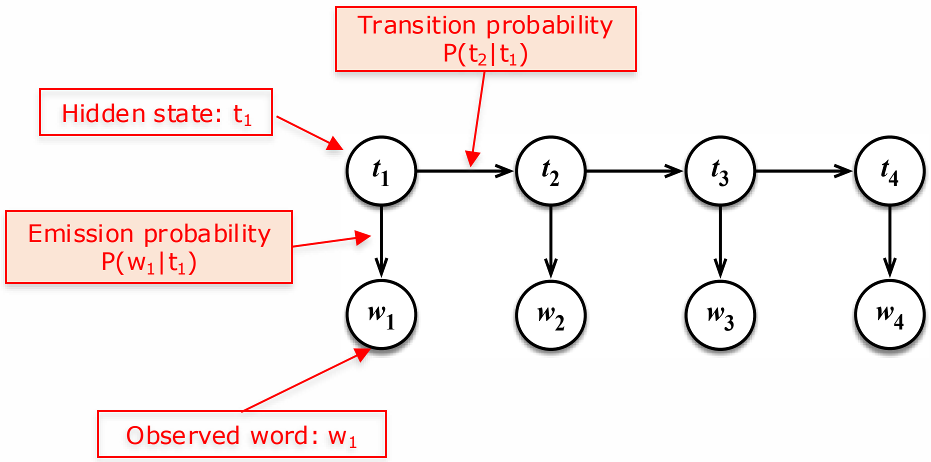


Note that:

- prediction for y_4 will depend on entire sequence $(w_1, w_2, \ldots, w_n)$, even subsequent terms (w_5)
- predicted values depend on each other and certain values of y_4 may not make sense with other $y_{i \neq 4}$ values.

How do sequence labellers work?

<u>Traditional methods</u> make use of either:



Hidden Markov Models (HMMs)

- ≈ Naïve Bayes applied to sequences
- which consist of:
  - unobserved states,
  - observed words,
  - transition probabilities linking states
  - emission probabilities for words in particular hidden states
- parameter estimation: simply count frequencies on hand labelled data, (use EM if hidden state unknown)

Conditional Random Fields (CRFs)

- ≈ Logistic Regression applied to sequences
- replaces transition and emission probabilities with **undirected potentials** $\phi(t_1,t_2)$ and $\phi(t_1,w_1)$
- get better performance by relaxing generative assumption, parameter estimation remains same

<u>Recent methods</u> make use of **Recurrent Neural Networks** (RNNs) to further improve performance

Recurrent Neural Networks

Aggregating embeddings

Word embeddings allow us to **represent words** in a semantic space ✔👍

How might we **aggregate embeddings** over words to represent a whole document?

- could simply add them up like we did with one-hot encodings to get a bag-of-words representation …
- but documents with **different word order** often have quite **different meaning:**
  - *there's a **white** rat in the house*
  - *there's a rat in the **white** house*

- and yet end up with the **same representation** …

one-hot vectors

there's		*a*		*white*		…		*house*		**bag-of-words** vector
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0
0		1		0				0		1
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0
0	+	0	+	0	+	…	+	0	=	0
1		0		0				1		2
0		0		0				0		0
0		0		0				0		0
0		0		1				0		2
0		0		0				0		0
:		:		:				:		:
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0
0		0		0				0		0

embedding vectors

there's		*a*		*white*		…		*house*		**sum of embedding** vectors
0.21		-0.32		-1.16				-0.11		0.31
-1.13		0.21		-1.13				-1.13		-1.13
0.15		-1.13		0.15				0.15		0.15
-0.32		0.15		-0.32				-0.32		-0.32
3.28	+	0.11	+	3.28	+	…	+	3.28	=	3.28
-0.33		-0.83		-0.33				-0.33		-0.33
0.15		3.28		9.15				0.15		9.15
0.11		-0.33		0.11				-2.11		0.11
-0.83		9.15		-0.83				0.68		-0.83
:		:		:				:		:
0.48		-0.80		0.48				0.81		0.38

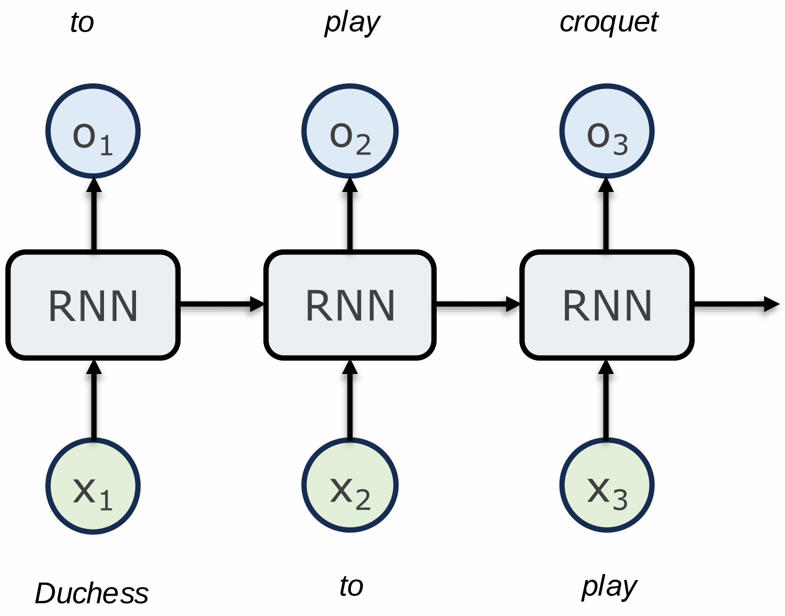
Recurrent Neural Networks (RNNs)

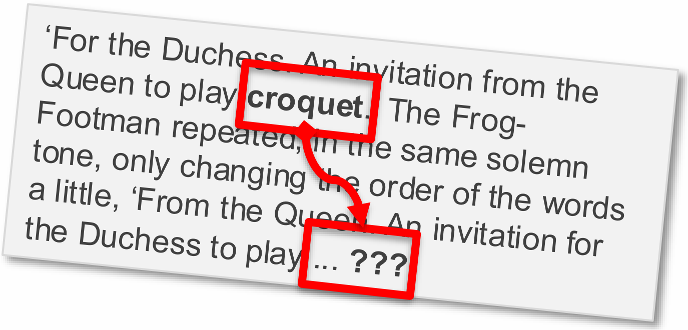
RNNs allow us to **aggregate information** over a document
- while **not ignoring** word order

RNNs provide general way to **accumulate information**
- by combining the **embedding** of current word
- with **context** from the previous words

RNNs are simply models which:
- take 2 vectors as input: <current input, previous state>
- produce 2 vectors as output: <current output, updated state>

They can be used to process **arbitrarily long** input contexts
- i.e. encode a sequence of text to a single embedding





Long Short-Term Memory (LSTM)

Clever implementation of RNN that is able to

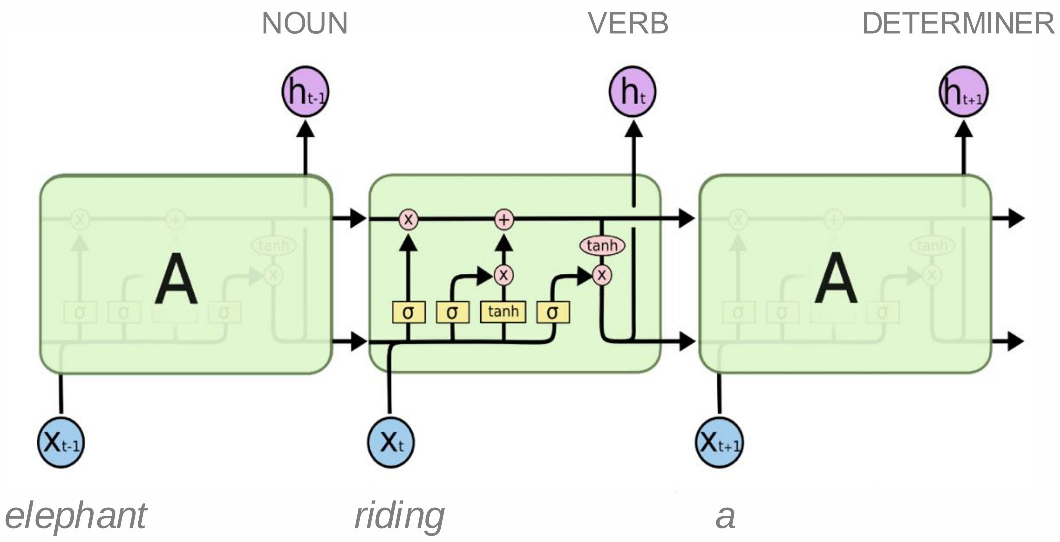
- learn **contexts** and **long-range dependencies**

Does this by using a **gating mechanism**

- passes through information by default
- unless new information is added to state
- or deleted from it (forgotten)

LSTM learns when & what information to

- **remember**, **forget**, and **output** at each timestep



Images source: Understanding LSTM Networks by Christopher Olah
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Aside: LSTMs and handling context

LSTMs can be stacked on top of each other

- have uncanny ability to handle nested contexts

Handling context is useful for natural language:

- for example, complete sentences with: *he/she/his/her*

  My mother was talking on the phone to ___ friend Jim.

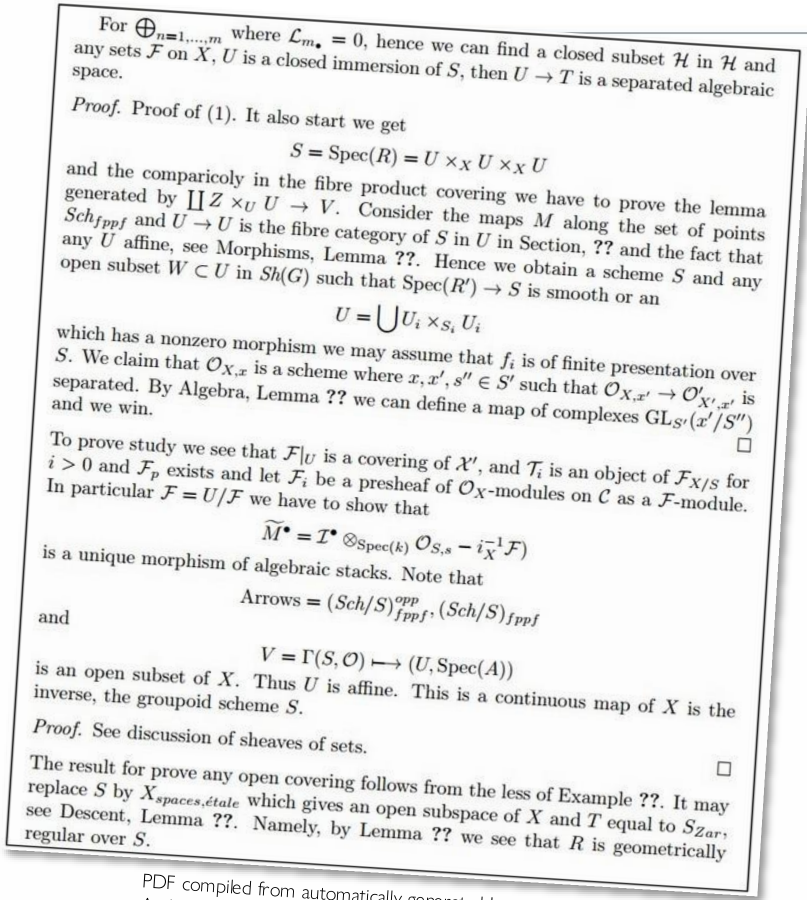
  Jim said that ___ favourite game was confusing ___ students.

  Replying, ___ said that ___ should find a better hobby.
- gender of subject changes for each subsequent sentence
- another example, this time with negation, complete with: *friendly/self-absorbed*

  I get along well with her brother. He's always ___

  I can not get along well with her brother. He's always ___

  I can not help but get along well with her brother. He's always ___
- LSTMs are able to switch between sentence and negation contexts



For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

PDF compiled from automatically generated Latex using multi-layer LSTM by Andrej Karpathy http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Applications

NLP Applications of Sequence Classifiers and Labellers

MANY applications of sequential models in NLP, including:

- part-of-speech tagging
- named entity extraction
- entity linkage
- relation extraction
- dependency parsing
- co-reference resolution
- …

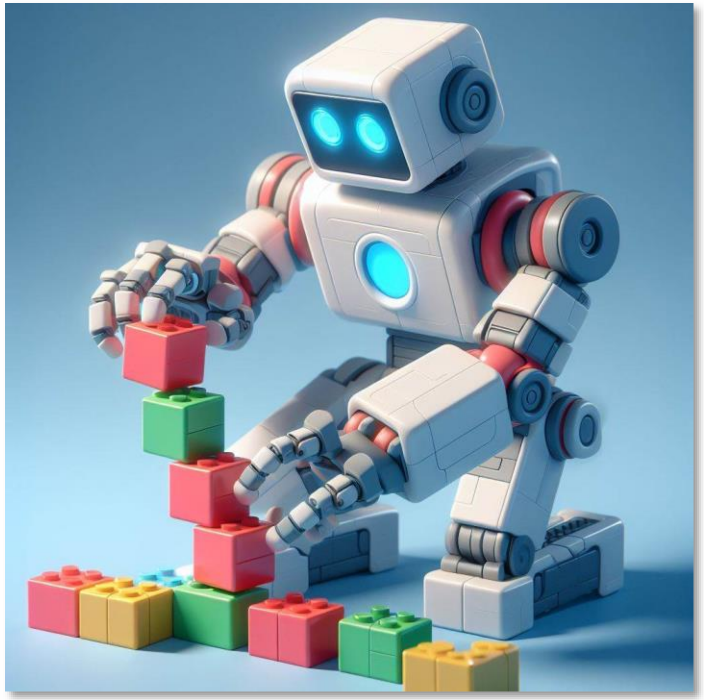


Image generated by "Microsoft Copilot | Designer" using keywords "**robot placing blocks in a row**"
https://www.bing.com/images/create/robot-placing-blocks-in-a-row/1-66139b53e13e4a57a1597ea954dcf53d?FORM=GUH2CR

part-of-speech (POS) tagging

Parts of Speech classes

Word classes have been around a **long time**:

- way back in the 1st century BCE, Dionysius Thrax of Alexandria defined: *nouns, verbs, pronouns, prepositions, adverbs, conjunctions, participles, articles*

Modern grammar divides world classes into open and closed:

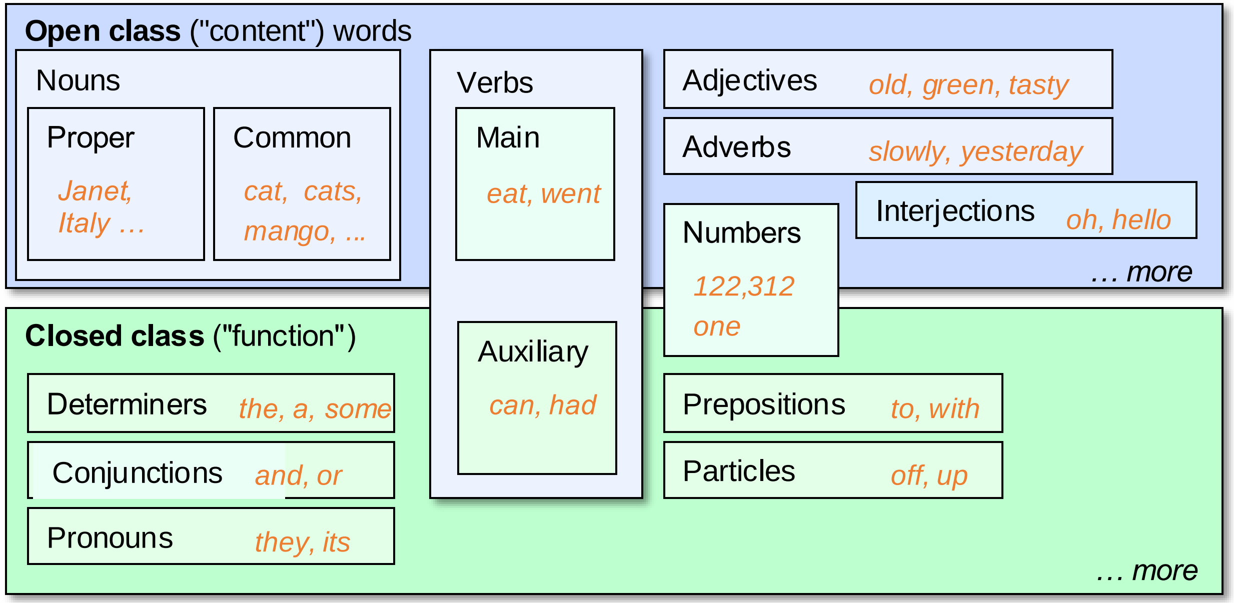
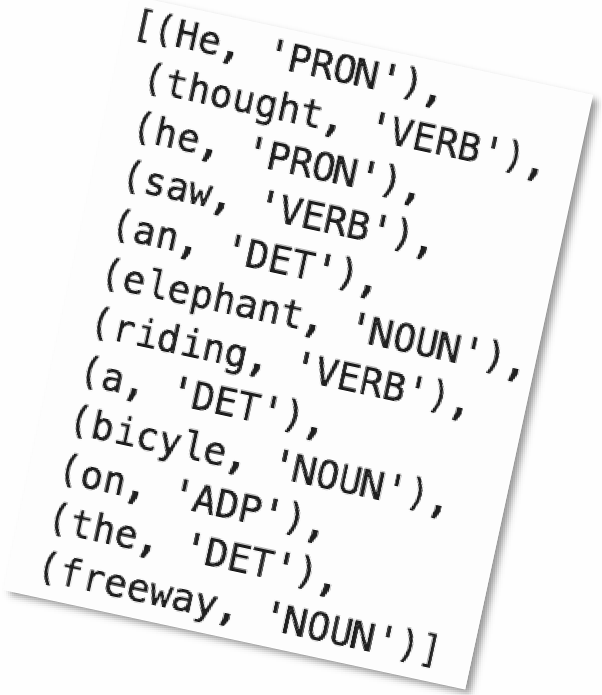


Image source: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition by Daniel Jurafsky and James H. Martin

What is POS tagging?



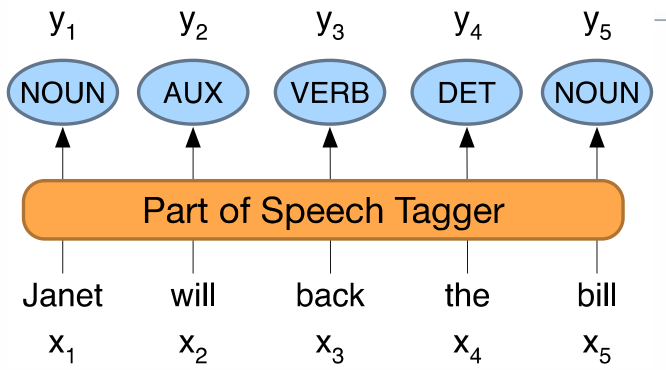
POS tagging:

- task of assigning to each token in a sequence:
- a **part-of-speech** label
- e.g.: PRON (pronoun), VERB, DET (determiner), NOUN, etc.

Why label parts-of-speech?

- useful for **developing features** for certain tasks
  - e.g. authorship attribution, particularly if only small amount of training data is available
- useful to **reduce ambiguity** in bag-of-words representation
  - some terms have different meaning depending on context "to **book**" vs "a **book**"
  - so append POS tag to each word occurrence: book_VERB vs book_NOUN
- useful as **initial step** for other NLP tasks or performing linguistic analysis
  - required for syntactic parsing
  - useful for text-to-speech
    - pronouncing "lead group" vs "lead weight" or "to object" vs "an object"
  - studying linguistic change like creation of new words, or meaning shift

Parts of Speech tagging



Map sequence of words $x_1, \ldots, x_n$ to sequence of POS tags $y_1, \ldots, y_n$

- set of tags:

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

- example sentences:
  - There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC
  - Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN 's/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

How difficult is POS tagging?

Approximately 85% of vocabulary terms in English are unambiguous

- *Janet* is always PROPN, *hesitantly* is always ADV

But ambiguous vocabulary terms are very common

- so ~60% of tokens are ambiguous

Example: word ***back*** could have 5 different POS tags:

- *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*

Accuracy of POS tagging is about 97%

- similar to human accuracy
- baseline (label each word with its most frequent tag) performance already 92%

Features used for POS tagging

Consider the example:

- `Janet will back the bill`

  AUX/NOUN/VERB? NOUN/VERB?

Sources of evidence for determingint the POS tags:

- Prior probabilities of word/tag
  - "will" is usually an AUX
- Identity of neighboring words
  - "the" means the next word is probably not a verb
- Morphology and wordshape:
  - Prefixes unable: un- → ADJ
  - Suffixes importantly: -ly → ADJ
  - Capitalization Janet: CAP → PROPN

named-entity-recognition (NER)

What is entity recognition?

Institution

"Have you taken any courses at the Politecnico di Milano taught by Mark Carman?"

Person

Named-Entity Recognition (NER):

- task of identifying entities that are mentioned in a text
- can be treated as a sequence labelling task
- often a first step in extracting knowledge from text

Named Entity Recognition (NER)

Named entity = object in real world

- most common tags:
  - PER (Person): e.g. "Marie Curie"
  - LOC (Location): e.g."Lake Michigan"
  - ORG (Organization): e.g. "Stanford University"
  - GPE (Geo-Political Entity): e.g. "Boulder, Colorado"
- often **multi-word phrases**
- term also extended to things that aren't entities:
  - dates, times, prices

Difference between a GPE and a LOC?

- *GPE : geopolitical entities, e.g. everything with a governing body like cities and countries. Examples: "Germany", "Buenos Aires".*
- *LOC : everything else that's a physical location or area, like "Kalahari Desert" or "Silicon Valley"*

Source: https://support.prodi.gy/t/ner-annotation-scheme-gpe-vs-loc/2913

NER task:

- find spans in text that constitute proper names
- and tags the type of the entity

Citing high fuel prices, [ORG **United Airlines**] said [TIME **Friday**] it has increased fares by [MONEY **$6**] per round trip on flights to some cities also served by lower-cost carriers. [ORG **American Airlines**], a unit of [ORG **AMR Corp.**], immediately matched the move, spokesman [PER **Tim Wagner**] said. [ORG **United**], a unit of [ORG **UAL Corp.**], said the increase took effect [TIME **Thursday**] and applies to most routes where it competes against discount carriers, such as [LOC **Chicago**] to [LOC **Dallas**] and [LOC **Denver**] to [LOC **San Francisco**].

Why NER?

Traditionally perform NER for:

- **Sentiment analysis**: identify sentiment towards particular company or person?
- **Information extraction**: extracting facts about entities from text
- **Question answering**: answer questions about an entity?
- **De-identification**: remove references to individual from text to protect privacy

NER can be hard because of:

1) **segmentation**: in POS tagging each word gets one tag, while in NER entities can be phrases
2) **type ambiguity**: same word/phrase can have many types depending on context

Person   Institution

Paris Hilton was photographed leaving the Paris Hilton.

Begin-Inside-Outside (BIO) Tagging

NER **finds phrases** in the text referring to named entities:

- *[PER Jane Villanueva] of [ORG United] , a unit of [ORG United Airlines Holding] , said the fare applies to the [LOC Chicago] route.*

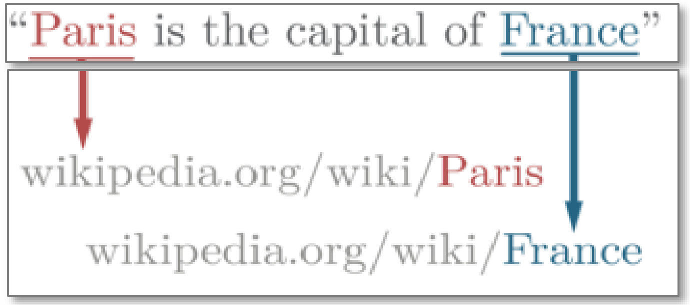
How can we turn NER into sequence labeling problem (with one label per token)?

- use begin/inside/outside tags:
  - B: token that **begins** a span
  - I: tokens **inside** a span
  - O: tokens **outside** of any span

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

Entity Linkage

What is entity linkage?



Source: https://commons.wikimedia.org/wiki/File:Entity_Linking_-_Short_Example.png

Determining that a named-entity has been mentioned in text

- often **only the first part** of problem
- second part: determine **which real-word entity** was referred to
- not as easy as it sounds!

I grew up in a small town just out of Paris.
Currently driving from Dallas to Paris.
My broken wrist is in a cast made from plaster of Paris.

Linkage techniques make use of:

- relative importance of entities
- context within text (other entities present)

Ontology/Knowledge Base

- generally **Wikipedia/DBPedia** is used
  - but many individuals/objects have no Wikipedia page
  - so custom custom ontologies can be used, particularly for medical tasks

Just had my photo taken with Michael Jordan!!
Just had my photo taken with Michael Jordan at EMNLP!!

Relation Extraction

What is relation extraction?

Source: https://commons.wikimedia.org/wiki/File:Entity_Linking_-_Short_Example.png

Once entity mentions have been linked to unique entities

- relationships between entities can be mined
- and used to populate a knowledge graph / knowledge base

Handled as a problem of predicting **missing links** in a graph

- entity embeddings can be leveraged for this purpose
  - since translations in space naturally encode relationships
  - see for example: https://arxiv.org/pdf/2002.00388.pdf

Mark Carman

Parse Trees

Parse Trees

1. $S \rightarrow aS$
2. $S \rightarrow bS$
3. $S \rightarrow a$
4. $S \rightarrow b$

Parse trees (also referred to as **syntax parse trees** or **dependency parse trees**)

- result from applying a **formal grammar** to analyze a sentence
- formal grammars define set of **rules** for **generating valid text**
  - see https://en.wikipedia.org/wiki/Formal_grammar
  - often used to define valid code for programming languages

Given piece of text, we can reverse the process (**parse** the text)

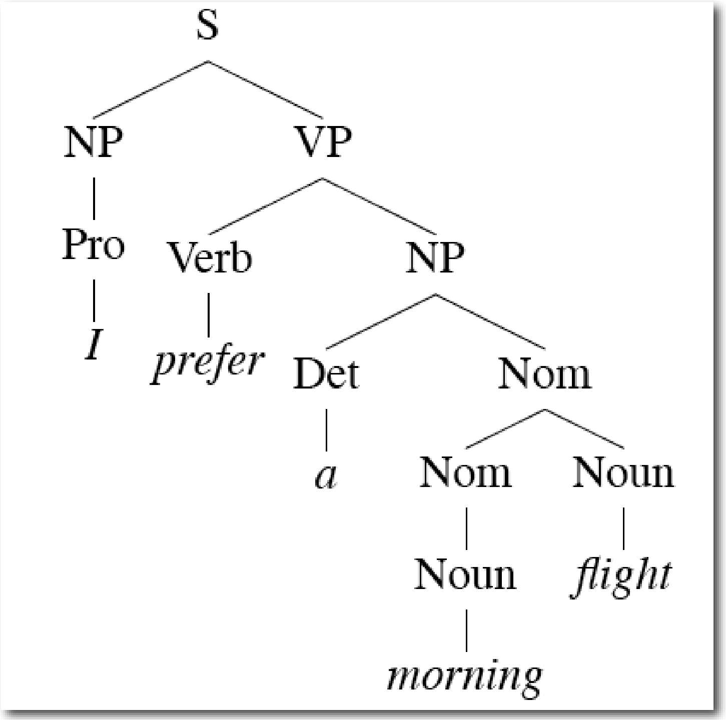
- to determine which rules have been applied, and in which order, to create it
- recursive application of rules results in **tree structure for each sentence**

Parse trees tell us:

- how the words in the sentence **relate to one another**
- from which we can try to **deduce intended meaning** (semantics) of the sentence

In theory, don't need machine learning for parsing text

- but in practice formal grammars are brittle and natural language can be ambiguous
- so need to use ML to extract parse tree

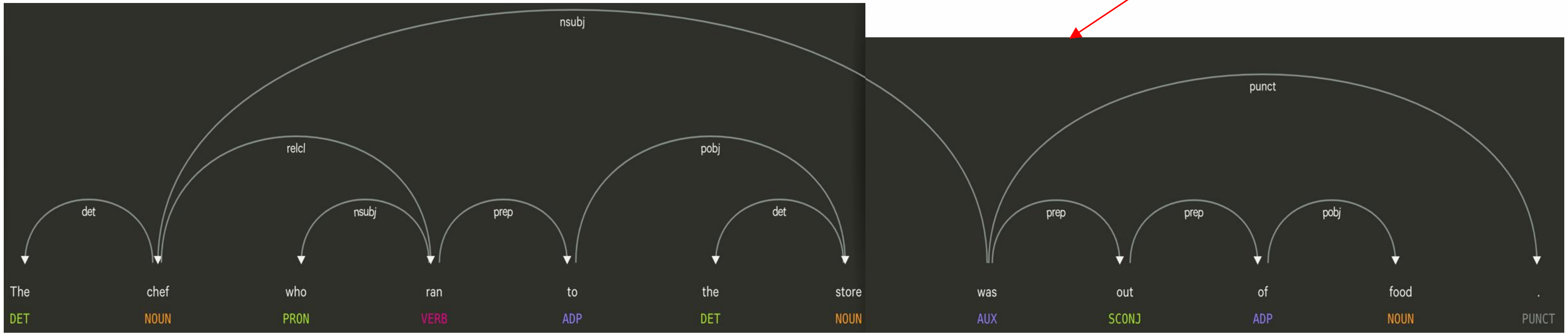


What are Parse Trees useful for?

For **understanding** the **meaning** of a sentence!

- consider the examples:
  - *The store was out of food.* and
  - *The chef who ran to the store was out of food.*
- may need to understand these sentences in order to:
  - **populate** a structured (SQL) **database** with information contained in them
  - generate a **valid next sentence** in the story that is consistent with them
- to do that, we will need to **know who did what** in the sentence
  - i.e. we need to who is out of food -- was it the store or was it the chef?
  - this relationship information is found in the parse tree:

In the second sentence it was the **chef** who *"was out of food"*, not the **store**



Penn Treebank

Famous dataset in which sentences are paired with their parse tree

- contains one million words from Wall Street Journal
- example sentence: *"We would have to wait until we have collected on those assets" he said.*
- corresponding parse tree in dataset:

```
( (S ('' '')
    (S-TPC-2
      (NP-SBJ-1 (PRP We) )
      (VP (MD would)
        (VP (VB have)
          (S
            (NP-SBJ (-NONE- *-1) )
            (VP (TO to)
              (VP (VB wait)
                (SBAR-TMP (IN until)
                  (S
                    (NP-SBJ (PRP we) )
                    (VP (VBP have)
                      (VP (VBN collected)
                        (PP-CLR (IN on)
                          (NP (DT those)(NNS assets)))))))))))))
    (, ,) ('' '')
    (NP-SBJ (PRP he) )
    (VP (VBD said)
      (S (-NONE- *T*-2) ))
    (. .) ))
```

Co-reference resolution

“No, ***I*** *could never vote for* ***Trump****.* ***He*** *doesn’t share* ***my*** *morals”*, **she** explained.

Co-reference Resolution

Problem of determining **who or what is being referenced** across (or sometimes within) sentences:

> ***John*** *went to* ***Bill's car dealership*** *to check out* ***an Acura Integra****.*
> ***He*** *looked at* ***it*** *for half an hour*

Who is **he**?

What is **it**?

In the second sentence:

- who is being referred to by the word ***he***?

  he refers to ***John***

- what is ***it***?

  is it ***Bill's car dealership*** or ***an Acura Integra***?

Co-reference Resolution - order

Regarding order of pronouns and referents

- most times the pronoun comes after the referent:

  John *went to the dealership to see a car that* ***he*** *was interested to purchase*

- but sometimes the pronoun comes before the referent:

  Before ***he*** *bought it,* ***John*** *checked over the Integra very carefully*

Why resolve co-references to entities from earlier/later in the text?

- to understand *what is being said about those entities*
  - (when pronouns are being used)
- needed for *information extraction* and *chatbots* …

Types of Reference Phenomena

Pronouns (*he, she, they, …*)

I saw no less than 6 Acura Integras today. They are the coolest cars.

A **non-pronominal** anaphora

one what?
one (of them)

I saw no less that 6 Acura Integra today. I want one.

Inferable anaphora

which engine?
the engine (of the Acura)

I almost bought an Acura Integra today, but the engine seemed noisy.

Demonstratives (*this, these, that, those*)

which one?
the one I bought five years ago

I bought an Integra yesterday, similar to the one I bought five years ago. That one was nice, but I like this one even more.

Demonstratives usually refer to entities, but other things can be referenced too:

According to John, Bob bought Sue an Integra, and Sue bought Fred a Legend

- *But that turned out to be a lie* (a speech act)
- *But that was false* (proposition)
- *That struck me as a funny way to describe the situation* (manner of description)
- *That caused Sue to become rather poor* (event)

Taxonomies and Ontologies

What are taxonomies and ontologies?

Taxonomy = hierarchy of concepts

- e.g. *types of products with is-a or part-of relationships*

Ontology = formal definition of concepts

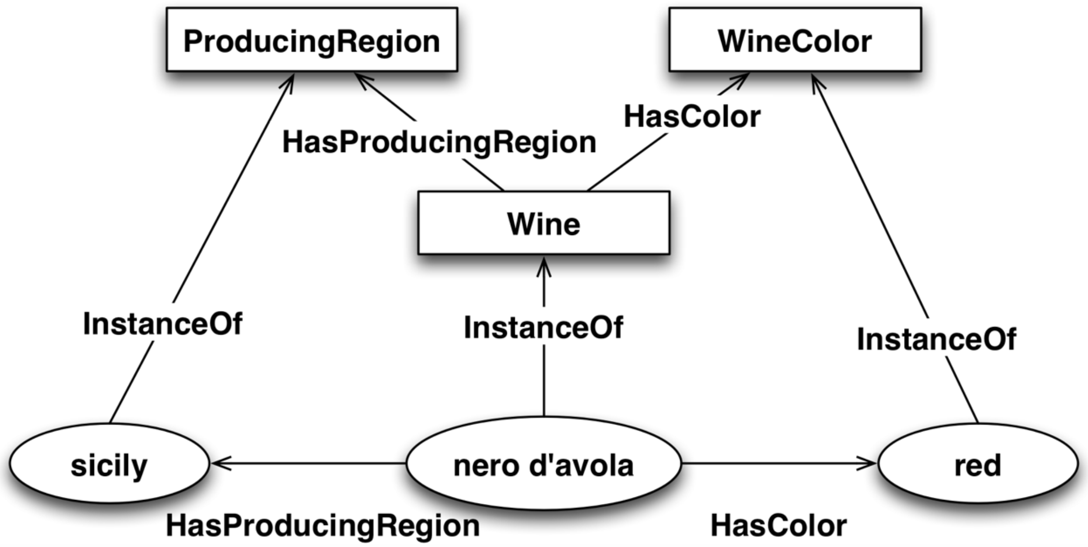
- abstract definition of concepts that *does not depend on the language*

Most ontologies are composed of:

- Classes: a set of objects / a type (e.g. **`wine`**)
- Individuals: an object (e.g. **`champagne`**)
- Attributes: property (e.g. **`price`**) with primitive data type (e.g. **`integer`**) allowing for restrictions on values (e.g., ">0")
- Relationships: characterization of relationships among classes or individuals (e.g. **`winery produces wine`**)
- Logical rules, e.g.: **`hasParent(?x1,?x2) ^ hasBrother(?x2,?x3) → hasUncle(?x1,?x3)`**

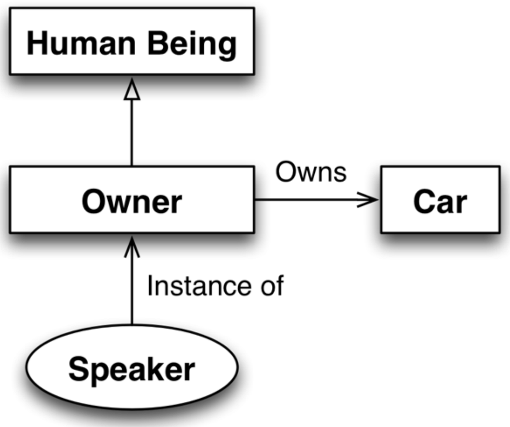
Ontologies as graphs

The relationships between concepts in an ontology/knowledge base form a graph:



Ontologies/knowledge bases can represent the information (*facts*) contained in sentences

- e.g. for the sentence *"I have a car"*



Knowledge Base Semantics

OWL (Web Ontology Language)

- based on RDF (triple: *subject predicate object*), i.e. Description Logic
- uses SPARQL query language to allow inference over KB such as DBPedia

Knowledge Bases have **open world** semantics:

- any statement that is *not known to be true* is *unknown*
  - as opposed to closed world assumption used in databases (SQL):
    any statement that is not known to be true is false (*negation as failure*)

Example if KB contains propositions: *"Giovanni is an architect"* and *"Giovanni is not a physicist"*

- query: *"Is Giovanni an engineer?"*
  - open world answer: **unknown**,
  - closed world answer: **no** (proposition not in KB)
- query: *"Is Giovanni a physicist?"*
  - open world answer: **no** (negated proposition found)
  - closed world answer: **no** (proposition not in KB)

Conclusions

Conclusions

The meaning of text depends on the **order of words**

- sequence classifiers take word order into account when categorizing text
- sequence labellers label each word in the sequence
- recurrent neural networks can be used to learn such models

Typical NLP tasks include

- part-of-speech tagging,
- named entity extraction and entity linkage
- relation extraction
- dependency parsing
- co-reference resolution