Why language is hard

And what Linguistics has to say about it

Natalia Silveira

Participation code: eagles

Language processing is so easy for humans that it is like trying to sell cargo airplanes to eagles.

Language processing is so easy for humans that it is like trying to sell cargo airplanes to eagles. They just don’t get what is hard, what is easy and the necessity of infrastructure. “Mr. Eagle, um, well we really need a runway to get the 20 tons of product into the air”.

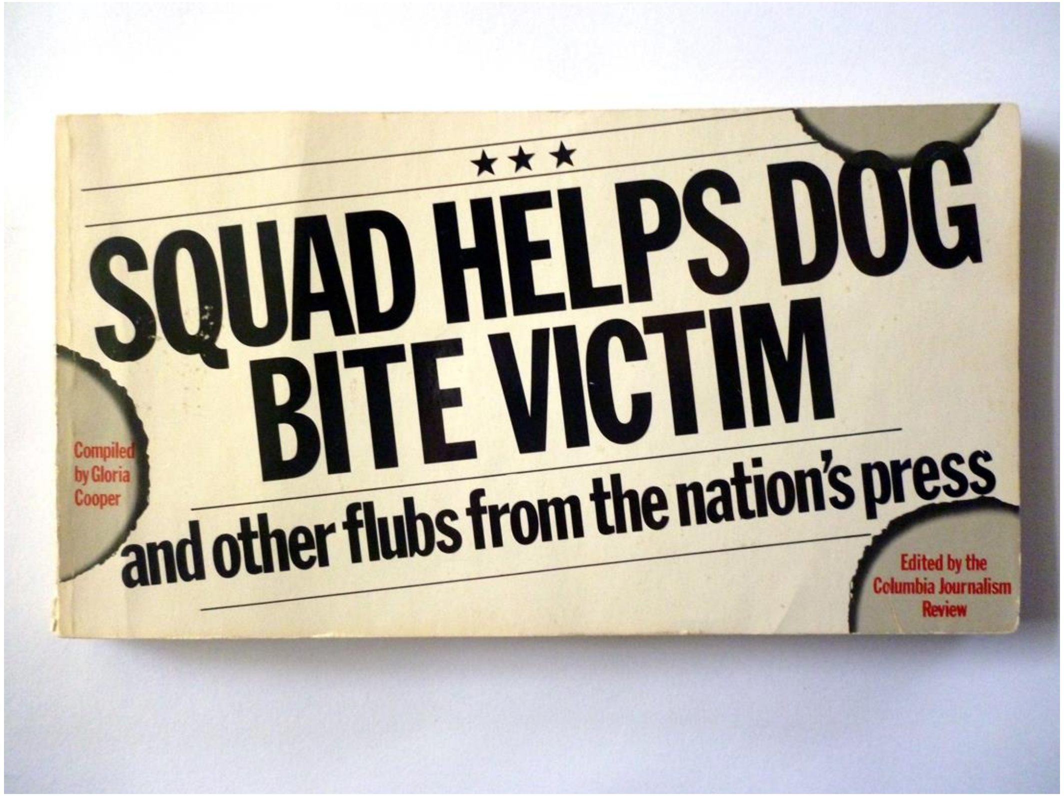
Language processing is so easy for humans that it is like trying to sell cargo airplanes to eagles. They just don’t get what is hard, what is easy and the necessity of infrastructure. “Mr. Eagle, um, well we really need a runway to get the 20 tons of product into the air”. Mr. Eagle responds with “What are you talking about? I can take off, land and raise a family on a tree branch. Cargo planes are easy because flying is easy for me. So I will give you a fish to do the job.”

Breck Baldwin in the LingPipe blog

Overview of today's lecture

- (A few reasons) Why language is hard
  - Exploring coarse processing vs. fine processing
  - In between: approaches to sentiment and difficult problems
- Intermission
- (A little bit of) What Linguistics has to say about it
  - Part-Of-Speech tagging
  - Dependency parsing

Language is multidimensional



Coarse vs. fine processing

- How do we do text classification?
- How do we do relation extraction?
- Why?

Coarse vs. fine processing

- How do we do text classification?
- How do we do relation extraction?
- Why?

Why language is hard: relation extraction

<PER>Bill Gates</PER> founded <ORG>Microsoft</ORG>

Why language is hard: relation extraction

Bill Gates founded Microsoft

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Microsoft , which Bill Gates founded, is a big software company.

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Microsoft , which Bill Gates founded, is a big software company.

Microsoft was founded by Bill Gates.

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Microsoft , which Bill Gates founded, is a big software company.

Microsoft was founded by Bill Gates.

Microsoft was founded not by Larry Page but by Bill Gates.

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Microsoft , which Bill Gates founded, is a big software company.

Microsoft was founded by Bill Gates.

Microsoft was founded not by Larry Page but by Bill Gates.

Microsoft was founded not only by Bill Gates but also Paul Allen.

Why language is hard: relation extraction

Bill Gates founded Microsoft

Bill Gates, now retired, founded the famous Microsoft

Microsoft , which Bill Gates founded, is a big software company.

Microsoft was founded by Bill Gates.

Microsoft was founded not by Larry Page but by Bill Gates.

Microsoft was founded not only by Bill Gates but also Paul Allen.

And that's not even the hard stuff!

Why language is hard: sentiment analysis

I liked this movie.

Why language is hard: sentiment analysis

I liked this movie.

I didn't like this movie.

Why language is hard: sentiment analysis

I liked this movie.

I didn't like this movie.

I thought I would like this movie.

Why language is hard: sentiment analysis

I liked this movie.

I didn't like this movie.

I thought I would like this movie.

I thought this movie would be great.

Why language is hard: sentiment analysis

I liked this movie.

I didn't like this movie.

I thought I would like this movie.

I thought this movie would be great.

I knew this movie would be great.

Why language is hard: sentiment analysis

I liked this movie.

I didn't like this movie.

I thought I would like this movie.

I thought this movie would be great.

I knew this movie would be great.

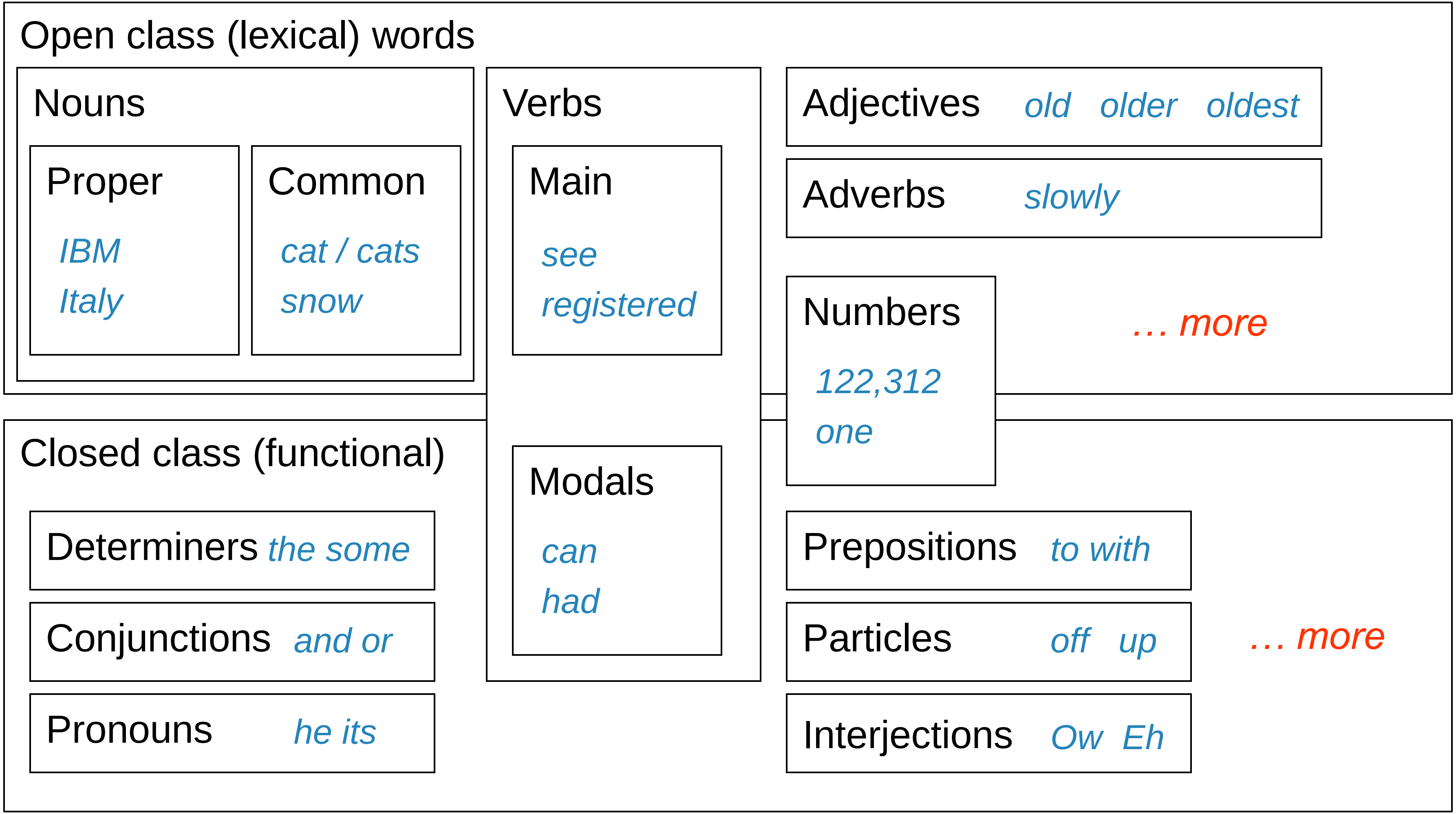
I didn't know this movie would be so great.

Part-of-speech tagging

A simple but useful form of linguistic analysis

Slides by Christopher Manning and Ray Mooney

Participation code: eagles



Open class (lexical) words
Nouns
Proper
IBM
Italy
Common
cat / cats
snow
Verbs
Main
see
registered
Adjectives
old older oldest
Adverbs
slowly
Numbers
122,312
one
… more
Closed class (functional)
Modals
can
had
Determiners the some
Conjunctions and or
Pronouns he its
Prepositions to with
Particles off up
Interjections Ow Eh
… more

Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
    - Why “closed”?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

POS Tagging

• Input:	Plays	well	with	others	
• Ambiguity:	NNS/VBZ	UH/JJ/NN/RB	IN	NNS	
• Output:	Plays/VBZ well/RB with/IN others/NNS				

Penn Treebank POS tags

- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - As input to or to speed up a full parser
  - If you know the tag, you can back off to it in other tasks

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!

Deciding on the correct part of speech can be difficult even for people

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

How difficult is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can't go *that* far = RB
- 40% of the word tokens are ambiguous

Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

More and Better Features → Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word the: the → DT
  - Lowercased word Importantly: importantly → RB
  - Prefixes unfathomable: un- → JJ
  - Suffixes Importantly: -ly → RB
  - Capitalization Meridian: CAP → NNP
  - Word shapes 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
  - Maxent P(t|w): 93.7% overall / 82.6% unknown

What else can we do?

- Build better features!

RB
PRP VBD IN RB IN PRP VBD .
They left as soon as he arrived .

- We could fix this with a feature that looked at the next word

JJ
NNP NNS VBD VBN .
Intrinsic flaws remained undetected .

- We could fix this by linking capitalized words to their lowercase versions

Sequence Labeling as Classification

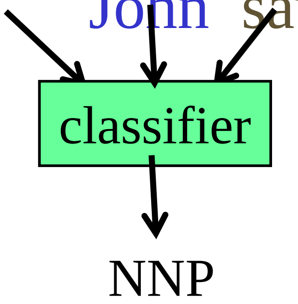
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.

Sequence Labeling as Classification

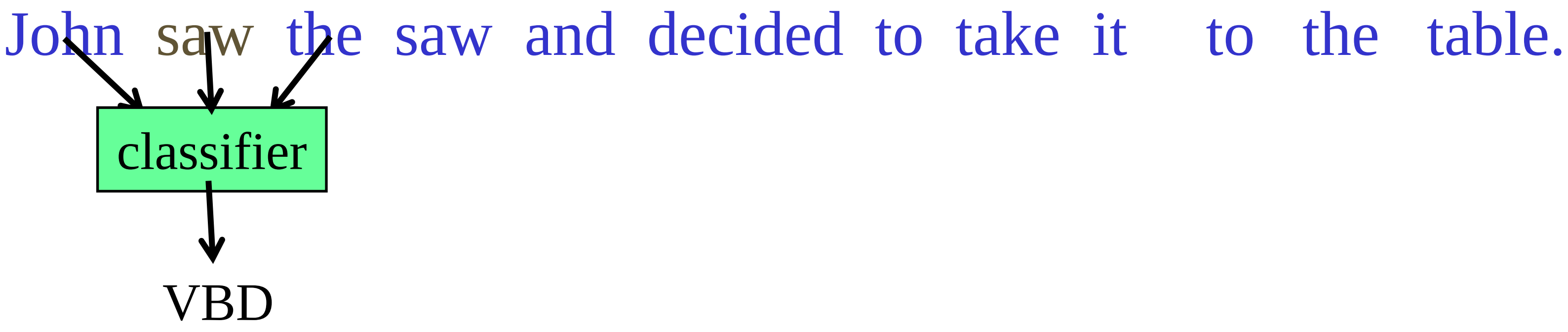
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



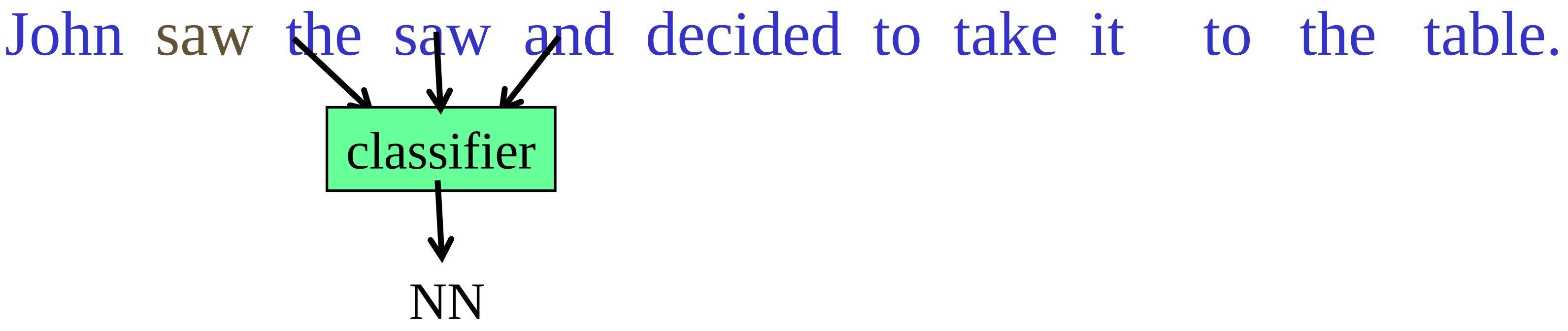
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



Sequence Labeling as Classification

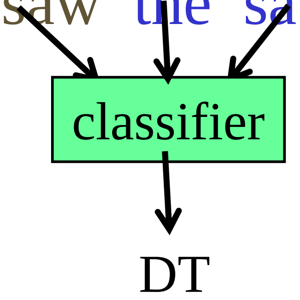
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



Sequence Labeling as Classification

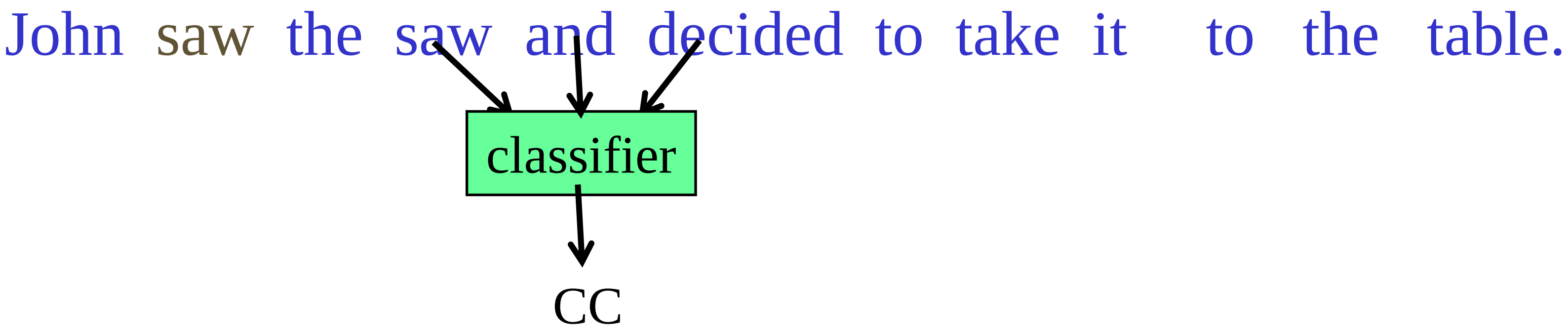
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



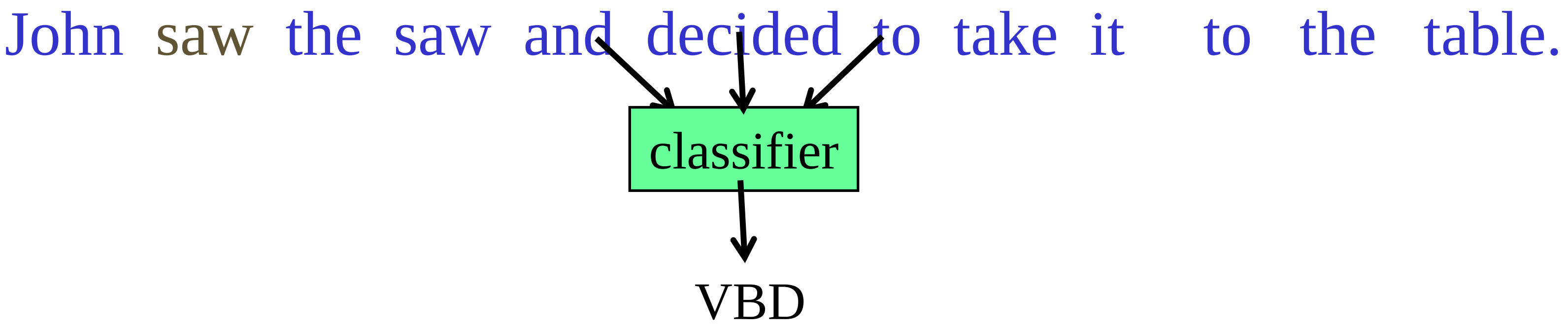
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



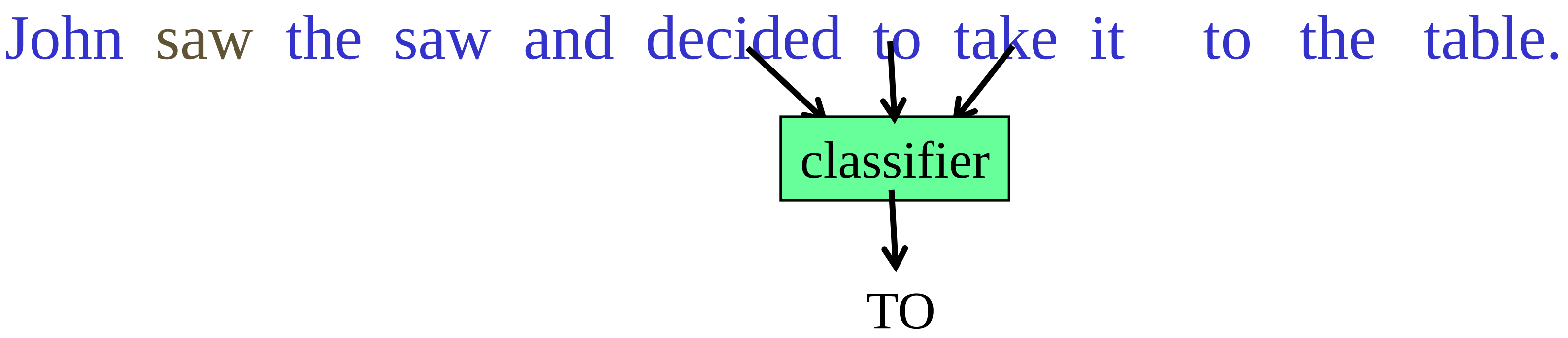
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



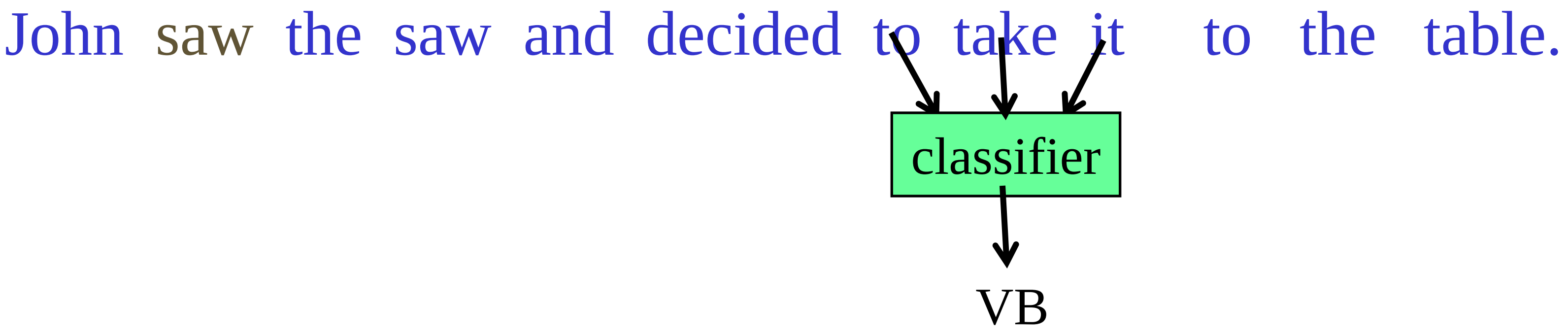
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



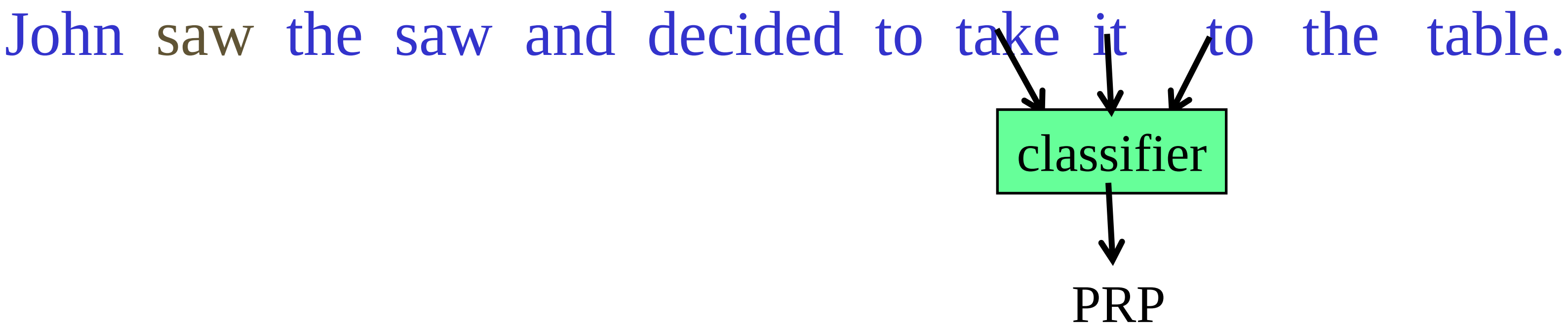
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



Sequence Labeling as Classification

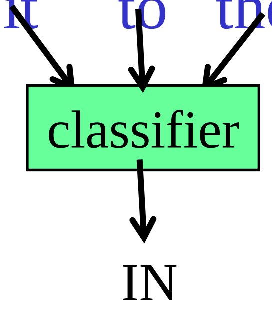
- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).



Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

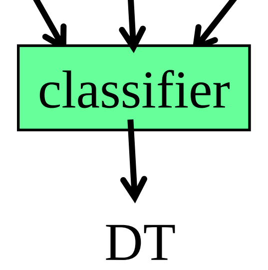
John saw the saw and decided to take it to the table.



Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

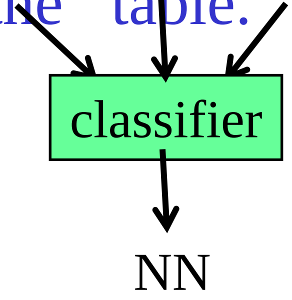
John saw the saw and decided to take it to the table.



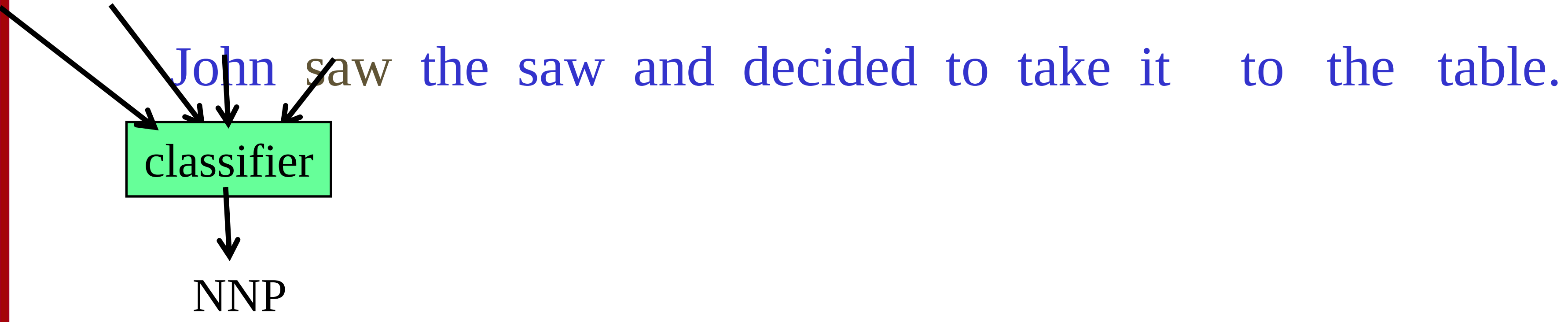
Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



Forward Classification



Forward Classification

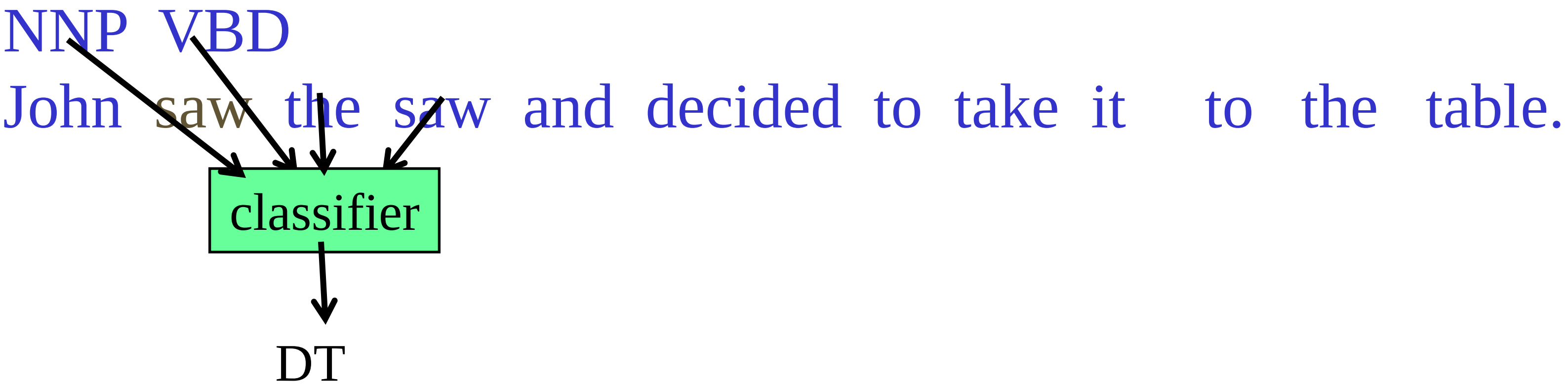
NNP

John saw the saw and decided to take it to the table.

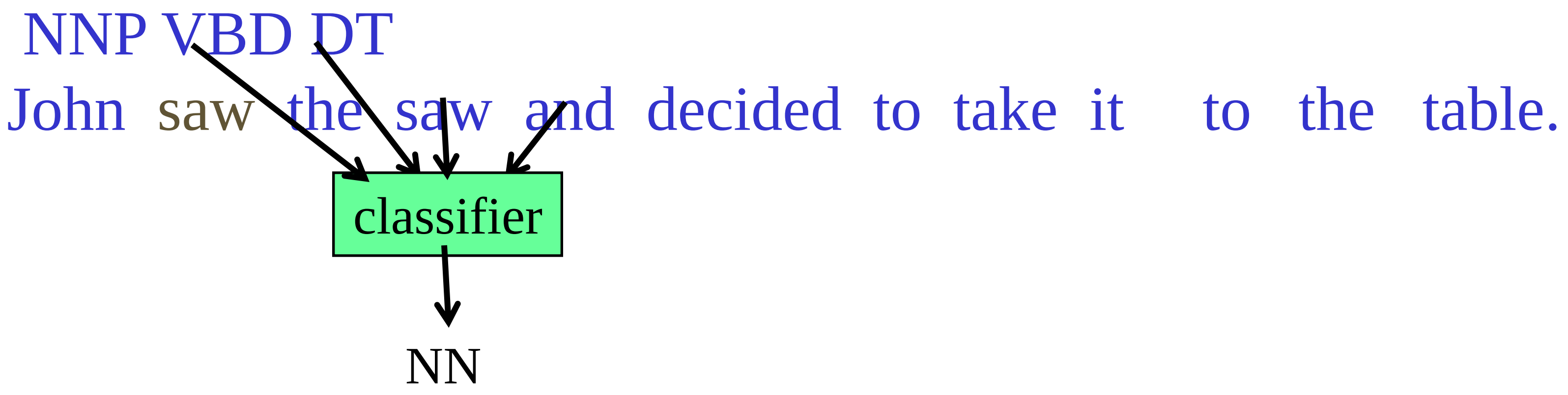
classifier

VBD

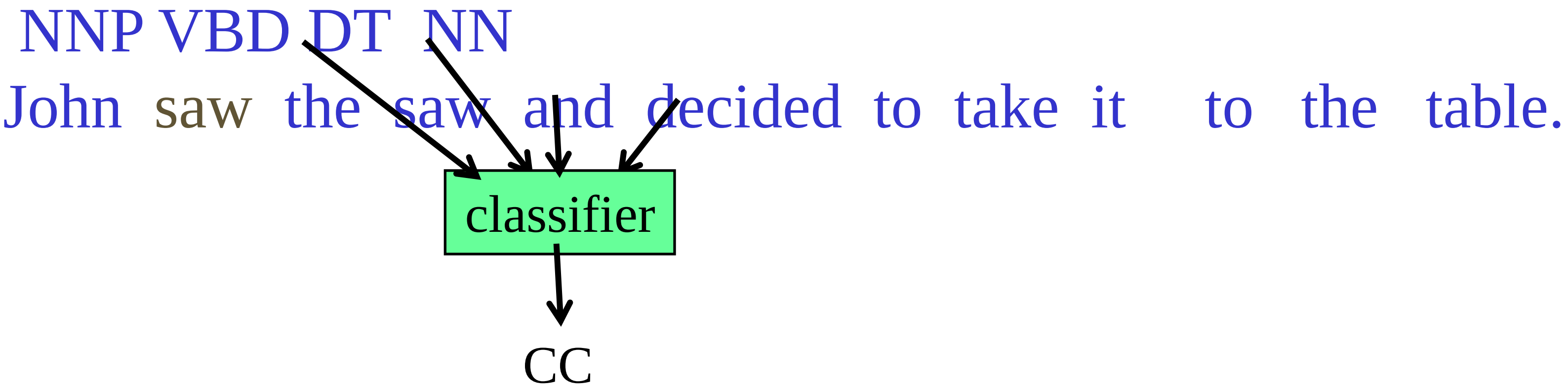
Forward Classification



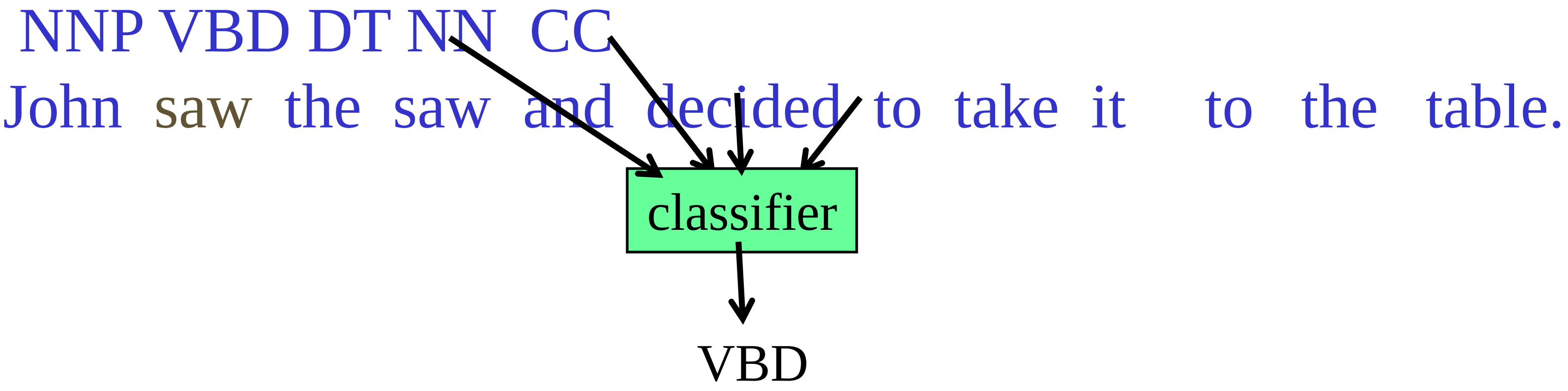
Forward Classification



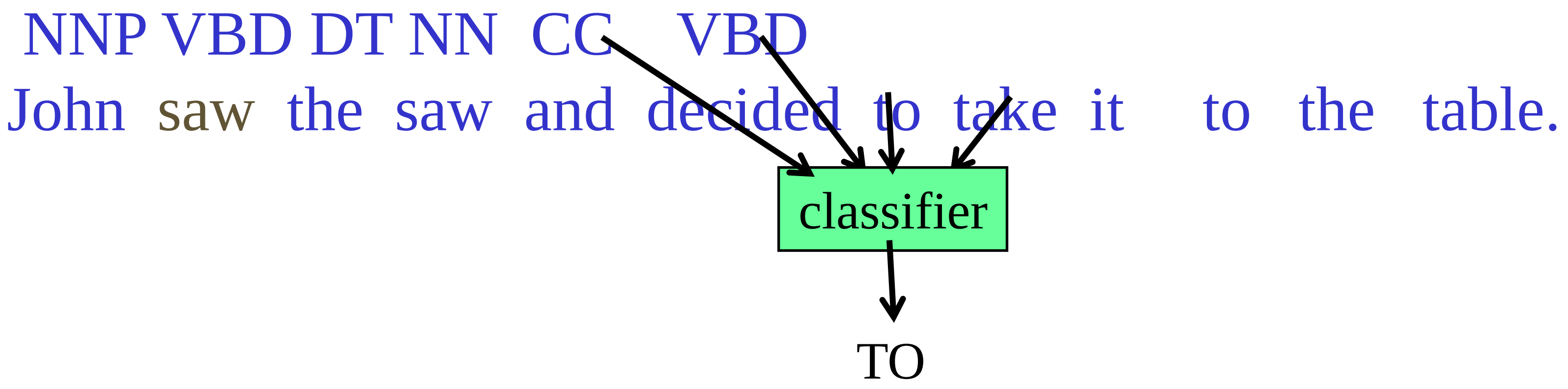
Forward Classification



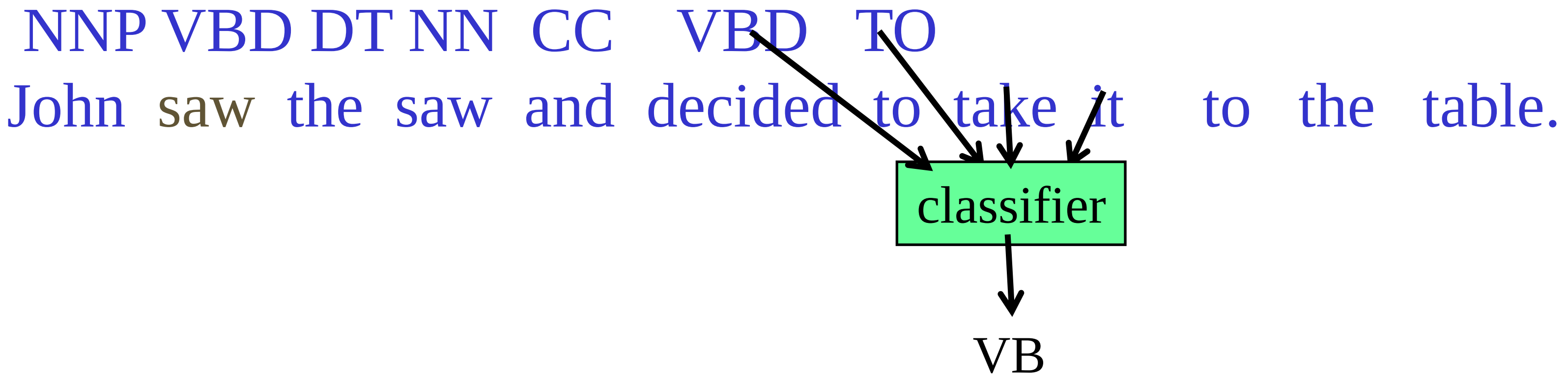
Forward Classification



Forward Classification



Forward Classification



Forward Classification

NNP VBD DT NN CC VBD TO VB

John saw the saw and decided to take it to the table.

classifier

PRP

Forward Classification

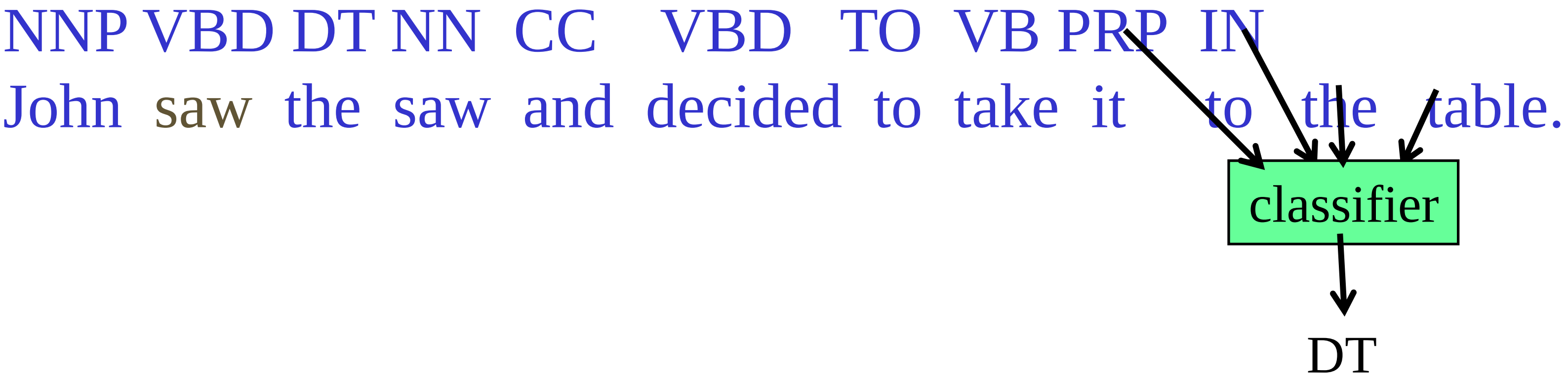
NNP VBD DT NN CC VBD TO VB PRP

John saw the saw and decided to take it to the table.

classifier

IN

Forward Classification



Forward Classification

NNP VBD DT NN CC VBD TO VB PRP IN DT

John saw the saw and decided to take it to the table.

classifier

NN

Backward Classification

- Disambiguating “to” in this case would be even easier backward.

John saw the saw and decided to take it to the table.

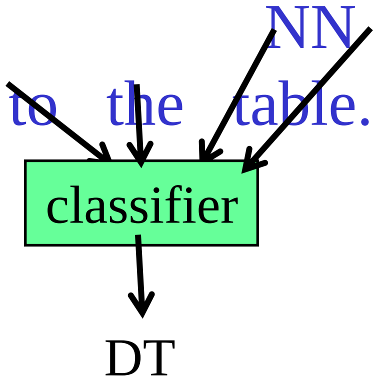
classifier

NN

Backward Classification

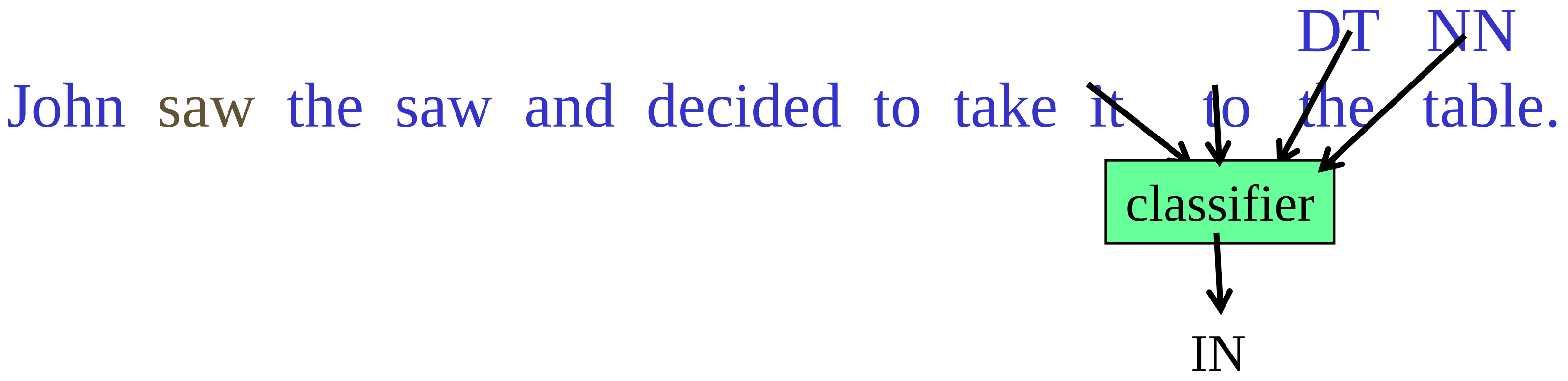
- Disambiguating "to" in this case would be even easier backward.

John saw the saw and decided to take it to the table.



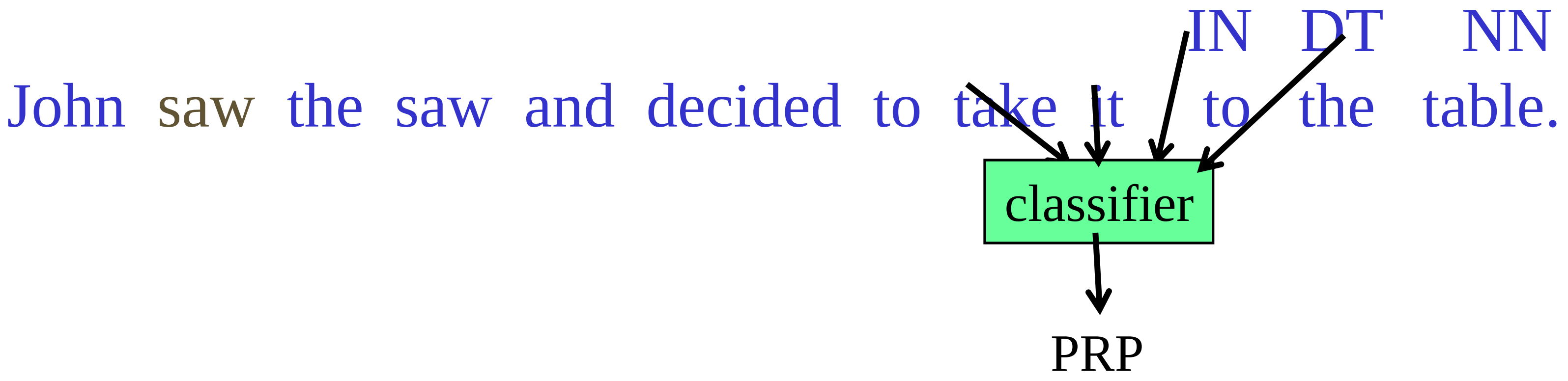
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



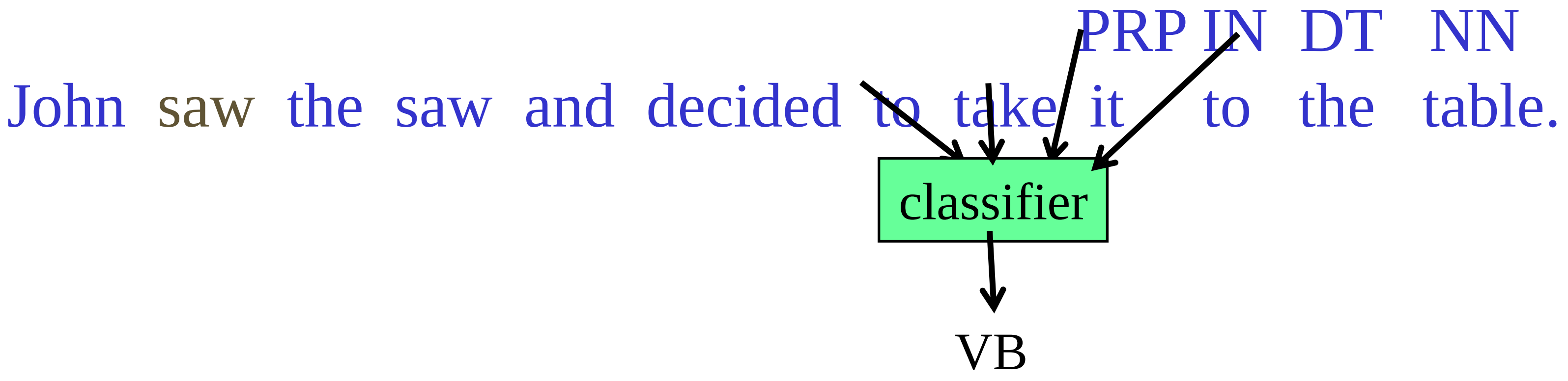
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



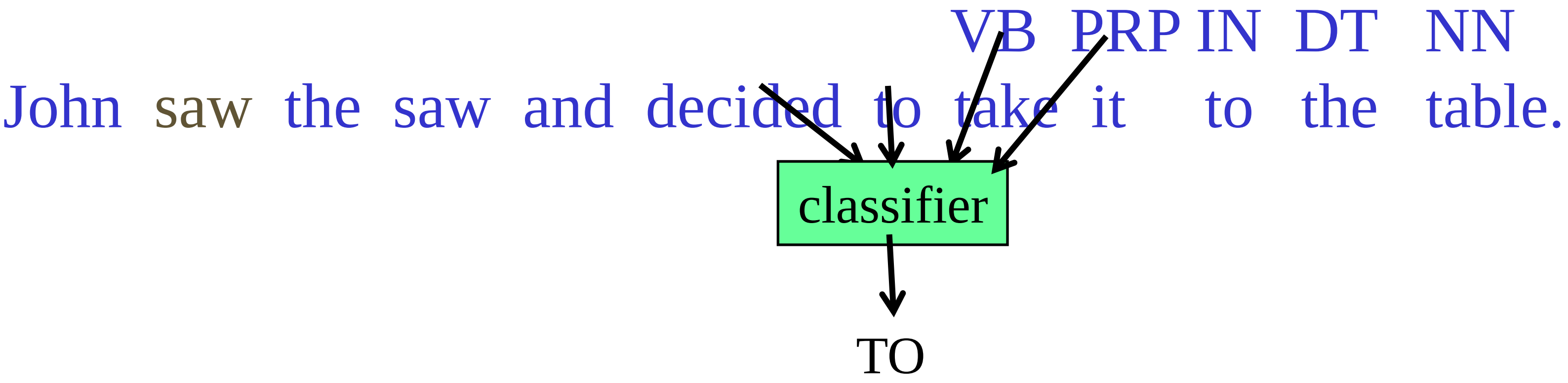
Backward Classification

- Disambiguating "to" in this case would be even easier backward.



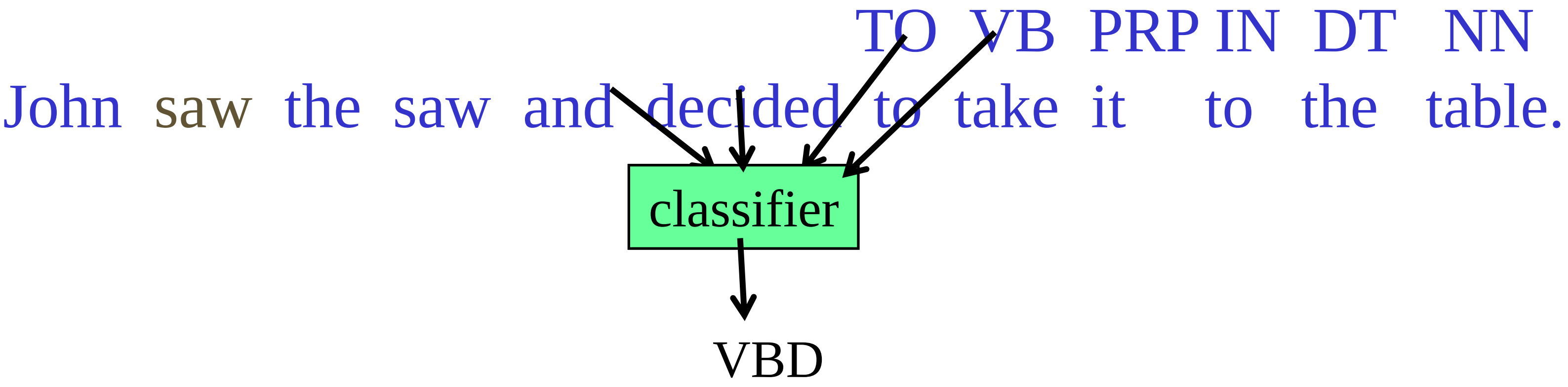
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



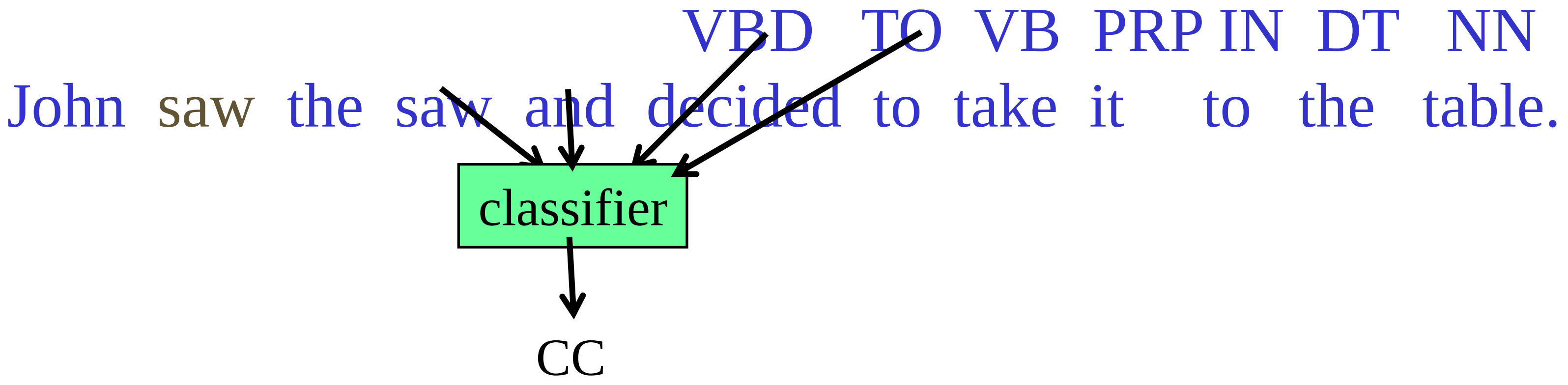
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



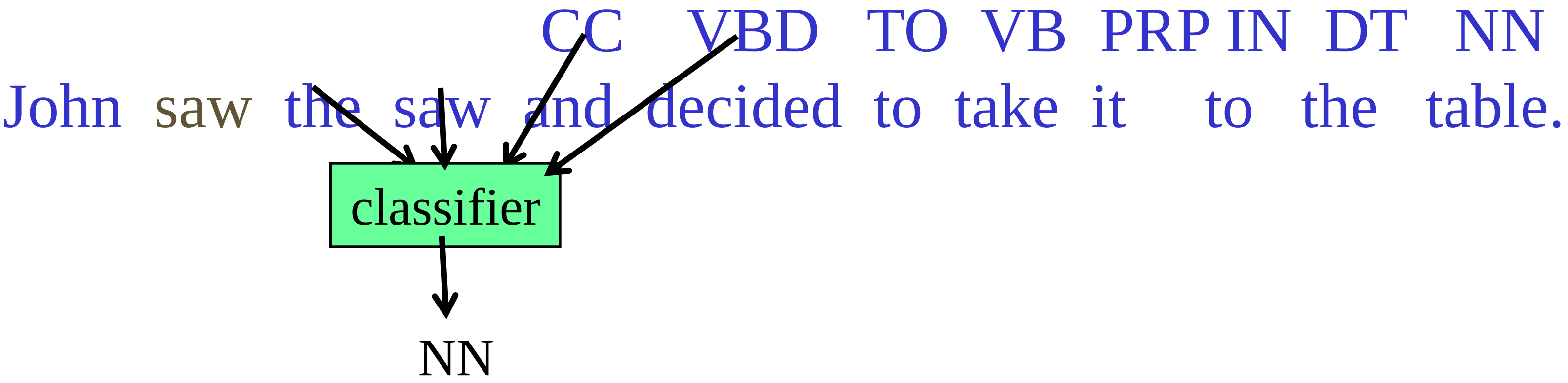
Backward Classification

- Disambiguating "to" in this case would be even easier backward.



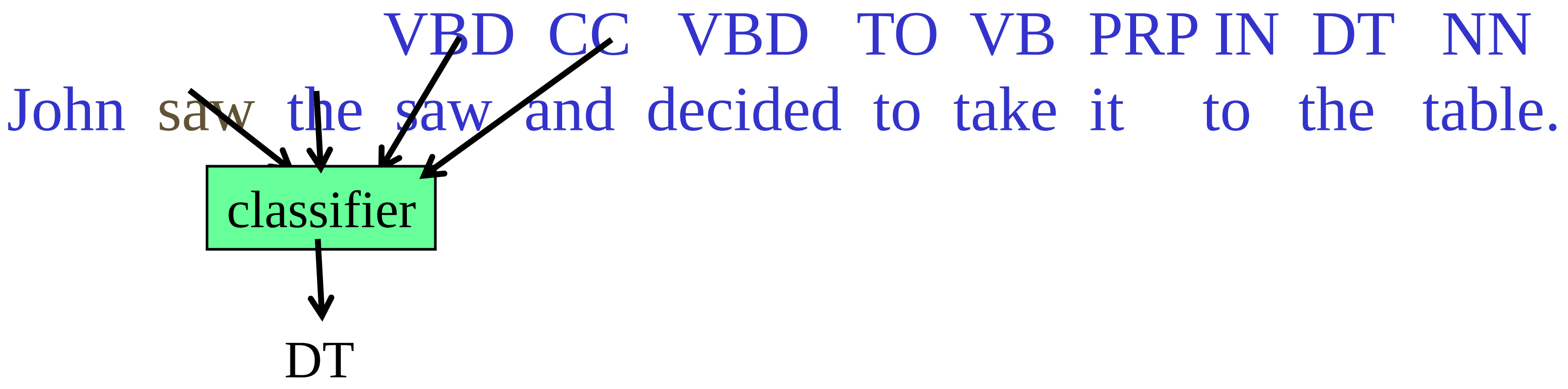
Backward Classification

- Disambiguating "to" in this case would be even easier backward.



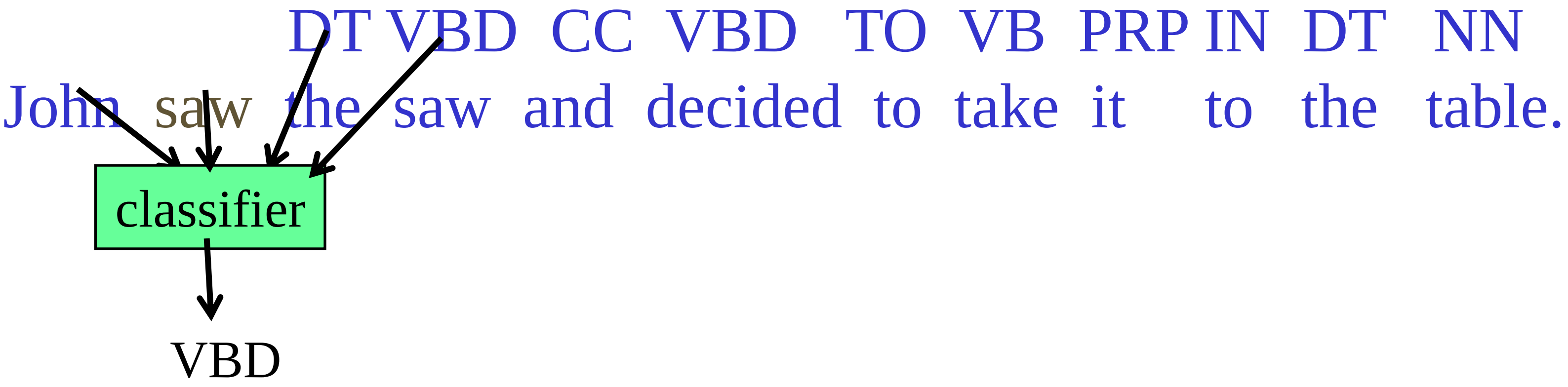
Backward Classification

- Disambiguating "to" in this case would be even easier backward.



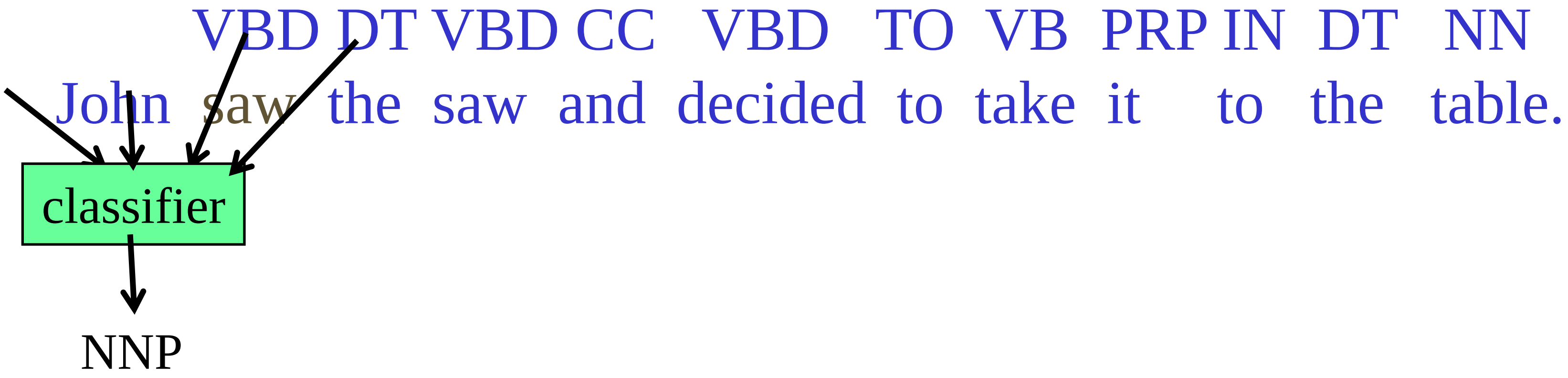
Backward Classification

- Disambiguating “to” in this case would be even easier backward.



Backward Classification

- Disambiguating "to" in this case would be even easier backward.



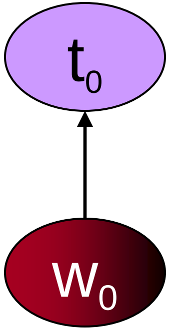
Overview: POS Tagging Accuracies

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Bidirectional dependencies: 97.2% / 90.0%
  - Upper bound: ~98% (human agreement)

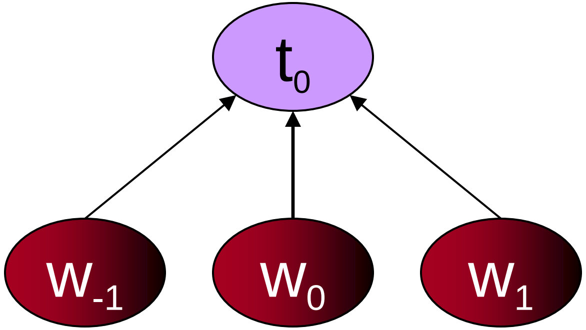
Most errors on unknown words

Tagging Without Sequence Information

Baseline



Three Words



Model	Features	Token	Unknown	Sentence
Baseline	56,805	**93.69%**	82.61%	26.74%
3Words	239,767	**96.57%**	86.78%	48.27%

Using words only in a straight classifier works as well as a basic (HMM or discriminative) sequence model!!

Summary of POS Tagging

For tagging, the change from generative to discriminative model **does not by itself** result in great improvement

One profits from models for specifying dependence on **overlapping features of the observation** such as spelling, suffix analysis, etc.

An MEMM allows integration of rich features of the observations, but can suffer strongly from assuming independence from following observations; this effect can be relieved by adding dependence on following words

This additional power (of the MEMM ,CRF, Perceptron models) has been shown to result in improvements in accuracy

The **higher accuracy** of discriminative models comes at the price of **much slower training**

Dependency Parsing

Building structure

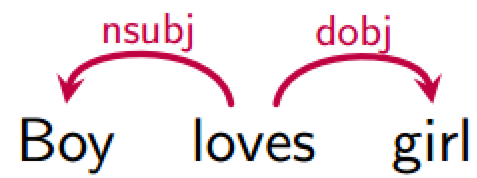
Based on slides by Christopher Manning

Participation code: eagles

Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies

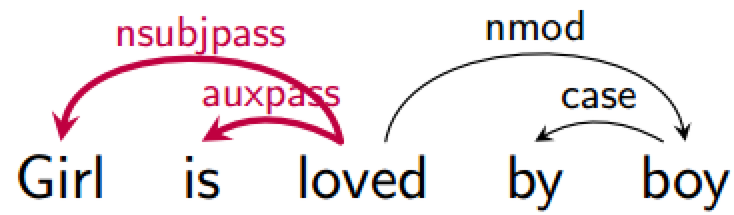
The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies

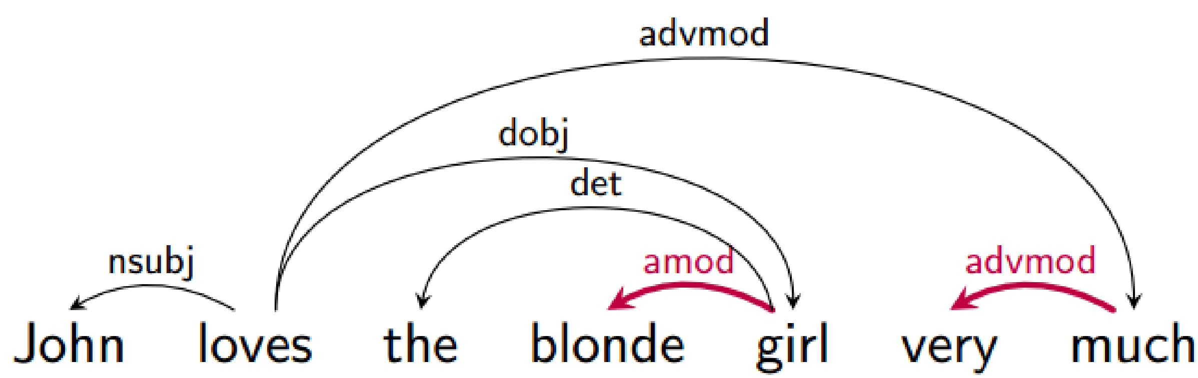
The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

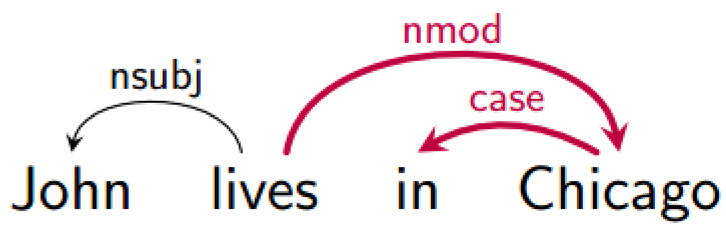
The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)

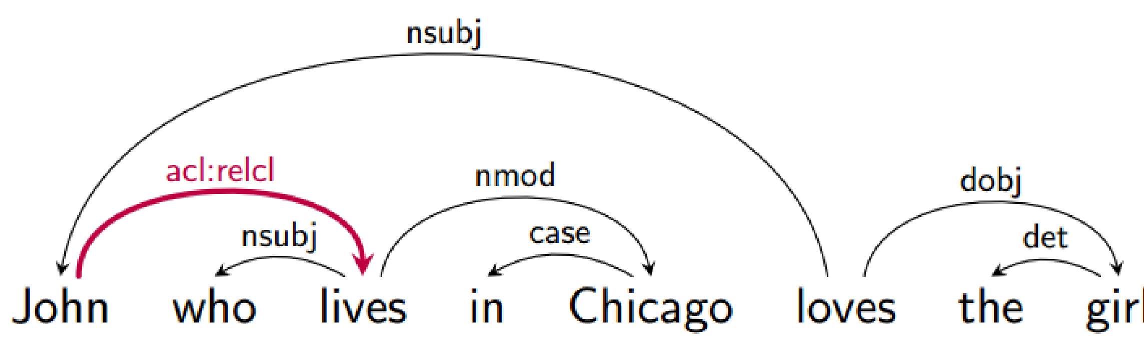


Christopher Manning

Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies

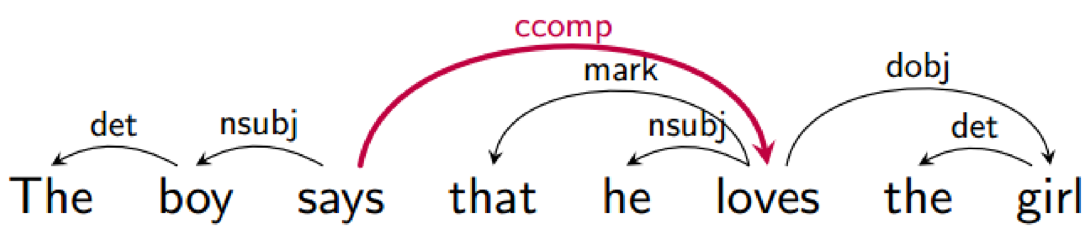
The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations (“arrows”) called dependencies

The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)



Methods of Dependency Parsing

1. Dynamic programming (like in the CKY algorithm)

   You can do it similarly to lexicalized PCFG parsing: an $O(n^5)$ algorithm

   Eisner (1996) gives a clever algorithm that reduces the complexity to $O(n^3)$, by producing parse items with heads at the ends rather than in the middle

2. Graph algorithms

   You create a Maximum Spanning Tree for a sentence

   McDonald et al.'s (2005) MSTParser scores dependencies independently using a ML classifier (he uses MIRA, for online learning, but it could be MaxEnt)

3. Constraint Satisfaction

   Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.

4. "Deterministic parsing"

   Greedy choice of attachments guided by machine learning classifiers

   MaltParser (Nivre et al. 2008) – discussed in the next segment

Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

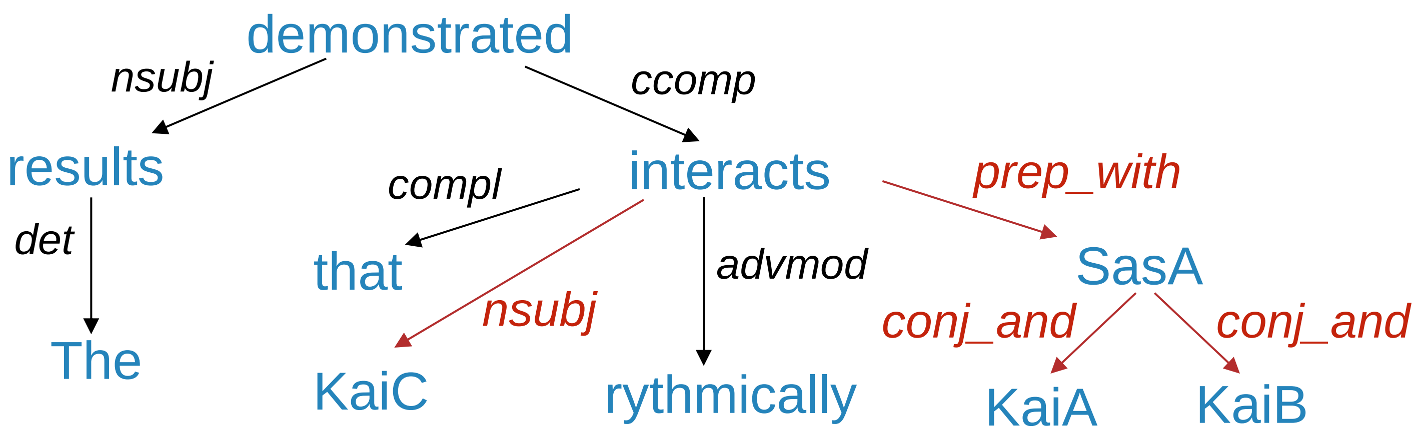
1. Bilexical affinities [girl → the] is plausible
2. POS tags [VB → NN] is plausible
3. Dependency distance mostly with nearby words
4. Intervening material

   Dependencies rarely span intervening verbs or punctuation
5. Valency of heads

   How many dependents on which side are usual for a head?

Dependency paths identify relations like protein interaction

[Erkan et al. EMNLP 07, Fundel et al. 2007]



KaiC ←nsubj interacts prep_with→ SasA

KaiC ←nsubj interacts prep_with→ SasA conj_and→ KaiA

KaiC ←nsubj interacts prep_with→ SasA conj_and→ KaiB

BioNLP 2009/2011 relation extraction shared tasks [Björne et al. 2009]