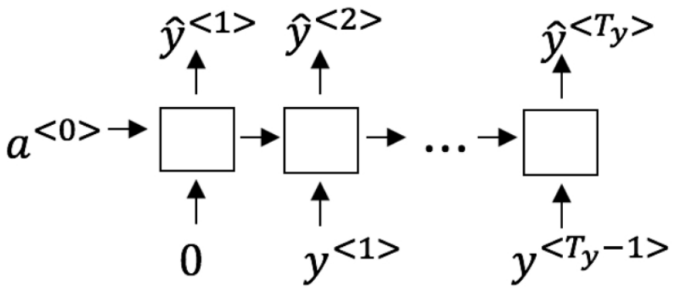
Correct

Correct!

4. You are training this RNN language model. 1 / 1 point



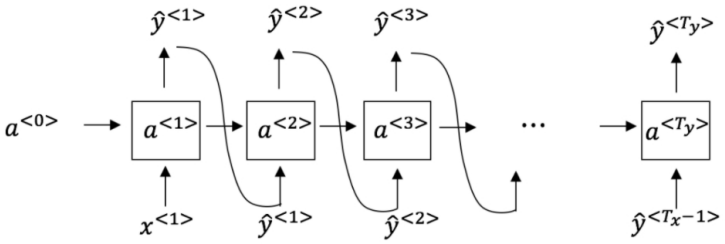
At the t^{th} time step, what is the RNN doing? Choose the best answer.

- Estimating $P(y^{<1>}, y^{<2>}, \ldots, y^{<t-1>})$
- Estimating $P(y^{<t>})$
- (selected) Estimating $P(y^{<t>} \mid y^{<1>}, y^{<2>}, \ldots, y^{<t-1>})$
- Estimating $P(y^{<t>} \mid y^{<1>}, y^{<2>}, \ldots, y^{<t>})$

Correct

Yes, in a language model we try to predict the next step based on the knowledge of all prior steps.

5. You have finished training a language model RNN and are using it to sample random sentences, as follows: 1 / 1 point



What are you doing at each time step t?

- (i) Use the probabilities output by the RNN to pick the highest probability word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass the ground-truth word from the training set to the next time-step.
- (i) Use the probabilities output by the RNN to randomly sample a chosen word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass the ground-truth word from the training set to the next time-step.
- (i) Use the probabilities output by the RNN to pick the highest probability word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass this selected word to the next time-step.
- (selected) (i) Use the probabilities output by the RNN to randomly sample a chosen word for that time-step as $\hat{y}^{<t>}$. (ii) Then pass this selected word to the next time-step.

Correct

Yes!

6. You are training an RNN, and find that your weights and activations are all taking on the value of NaN ("Not a Number"). Which of these is the most likely cause of this problem? 1 / 1 point

- Vanishing gradient problem.
- (selected) Exploding gradient problem.
- ReLU activation function g(.) used to compute g(z), where z is too large.
- Sigmoid activation function g(.) used to compute g(z), where z is too large.

Correct

7. Suppose you are training a LSTM. You have a 10000 word vocabulary, and are using an LSTM with 100-dimensional activations $a^{<t>}$. What is the dimension of Γ_u at each time step? 1 / 1 point

- 1
- (selected) 100
- 300
- 10000

✓ **Correct**

Correct, Γ_u is a vector of dimension equal to the number of hidden units in the LSTM.

8. Here're the update equations for the GRU. **1 / 1 point**

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

Alice proposes to simplify the GRU by always removing the Γ_u. I.e., setting Γ_u = 1. Betty proposes to simplify the GRU by removing the Γ_r. I. e., setting Γ_r = 1 always. Which of these models is more likely to work without vanishing gradient problems even when trained on very long input sequences?

- ○ Alice's model (removing Γ_u), because if $\Gamma_r \approx 0$ for a timestep, the gradient can propagate back through that timestep without much decay.
- ○ Alice's model (removing Γ_u), because if $\Gamma_r \approx 1$ for a timestep, the gradient can propagate back through that timestep without much decay.
- ◉ Betty's model (removing Γ_r), because if $\Gamma_u \approx 0$ for a timestep, the gradient can propagate back through that timestep without much decay.
- ○ Betty's model (removing Γ_r), because if $\Gamma_u \approx 1$ for a timestep, the gradient can propagate back through that timestep without much decay.

✓ **Correct**

Yes. For the signal to backpropagate without vanishing, we need $c^{<t>}$ to be highly dependant on $c^{<t-1>}$.

9. Here are the equations for the GRU and the LSTM: **1 / 1 point**

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

From these, we can see that the Update Gate and Forget Gate in the LSTM play a role similar to _____ and _____ in the GRU. What should go in the the blanks?

- ◉ Γ_u and $1 - \Gamma_u$
- ○ Γ_u and Γ_r
- ○ $1 - \Gamma_u$ and Γ_u
- ○ Γ_r and Γ_u

✓ **Correct**

Yes, correct!

10. You have a pet dog whose mood is heavily dependent on the current and past few days' weather. You've collected data for the past 365 days on the weather, which you represent as a sequence as $x^{<1>}, \ldots, x^{<365>}$. You've also collected data on your dog's mood, which you represent as $y^{<1>}, \ldots, y^{<365>}$. You'd like to build a model to map from $x \rightarrow y$. Should you use a Unidirectional RNN or Bidirectional RNN for this problem? **1 / 1 point**

- ○ Bidirectional RNN, because this allows the prediction of mood on day t to take into account more information.
- ○ Bidirectional RNN, because this allows backpropagation to compute more accurate gradients.
- ◉ Unidirectional RNN, because the value of $y^{<t>}$ depends only on $x^{<1>}, \ldots, x^{<t>}$, but not on $x^{<t+1>}, \ldots, x^{<365>}$
- ○ Unidirectional RNN, because the value of $y^{<t>}$ depends only on $x^{<t>}$, and not other days' weather.

✓ **Correct**

Yes!

How many trigrams phrases can be generated from the following sentence, after replacing punctuations by a single space?
<Natural Language processing is very interesting, though not easy.=
Solution: (Any one from 2 options correct)
Number of trigrams=8
<s> Natural Language, Natural Language processing, Language processing is, processing is very, is very interesting, very interesting though, interesting though not, though not easy
OR
Number of trigrams=9
<s> Natural Language, Natural Language processing, Language processing is, processing is very, is very interesting, very interesting though, interesting though not, though not easy, not easy </s>

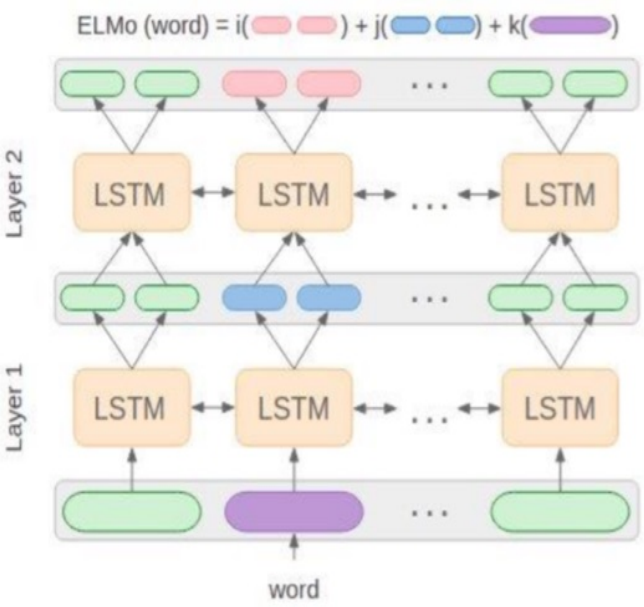
Write the formulae to calculate the unigram, bigram and trigram probabilities of the below sentence
"Life should be great rather than long"
Solution:
Unigram
P (<Life should be great rather than long)
=P(Life)P(should)P(be)P(great)P(rather)P(than)P(long)

Bigram
P (<Life should be great rather than long)
=P(Life|<s>)P(should|Life))P(be|should)P(great|be)P(rather|great)P(than|rather)P(long|than)

Trigram
P (<Life should be great rather than long=)
=P(Life|<s>,<S>)P(should|Life,<s>))P(be|should,life)P(great|be, should)P(rather|great,be)
P(than|rather,great)P(long|than,rather)

ELMO Embeddings from Language Model

Embeddings from Language Models (ELMO)

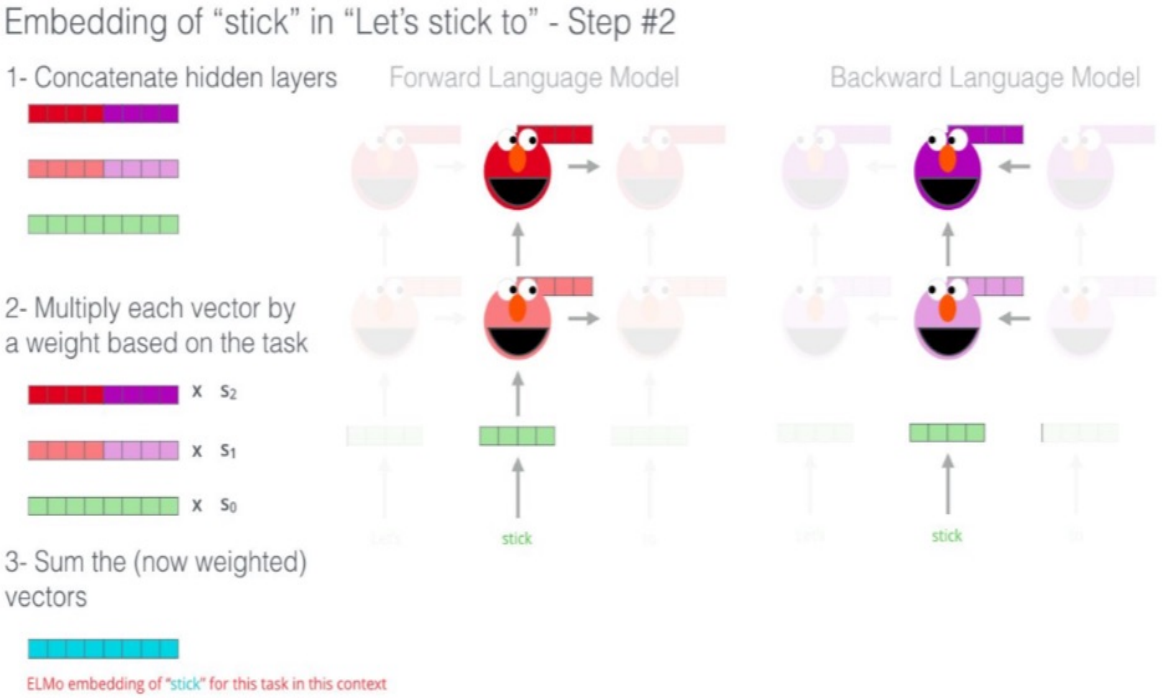


- Stacked LSTMs
- Each LSTM layer i gives a representation h_i of the token t_i
- Final representation h is a combination of the representations from different layers
  - $h=f(h_0,h_1,\ldots,h_L)$
- How to combine these representations?
- How to use in target tasks?

Ref: Peters, Matthew E., et al. "Deep contextualized word representations." NAACL 2018.

Figure from: Biesialska, K.et al. (2020). Sentiment analysis with contextual embeddings and self-attention. In *International Symposium on Methodologies for Intelligent Systems* (pp. 32-41).

Combining Representations from ELMO Layers



Picture courtesy: The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning) by Jay Alammar

- Embeddings from Language Models: **ELMo**
- Learn word embeddings through building *bidirectional language models* (biLMs)
  - biLMs consist of forward and backward LMs
    - Forward: $p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$
    - Backward: $p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N)$

A biLM combines both forward & backward LM. our formulation jointly maximizes the log likelihood of the forward & backward direction:

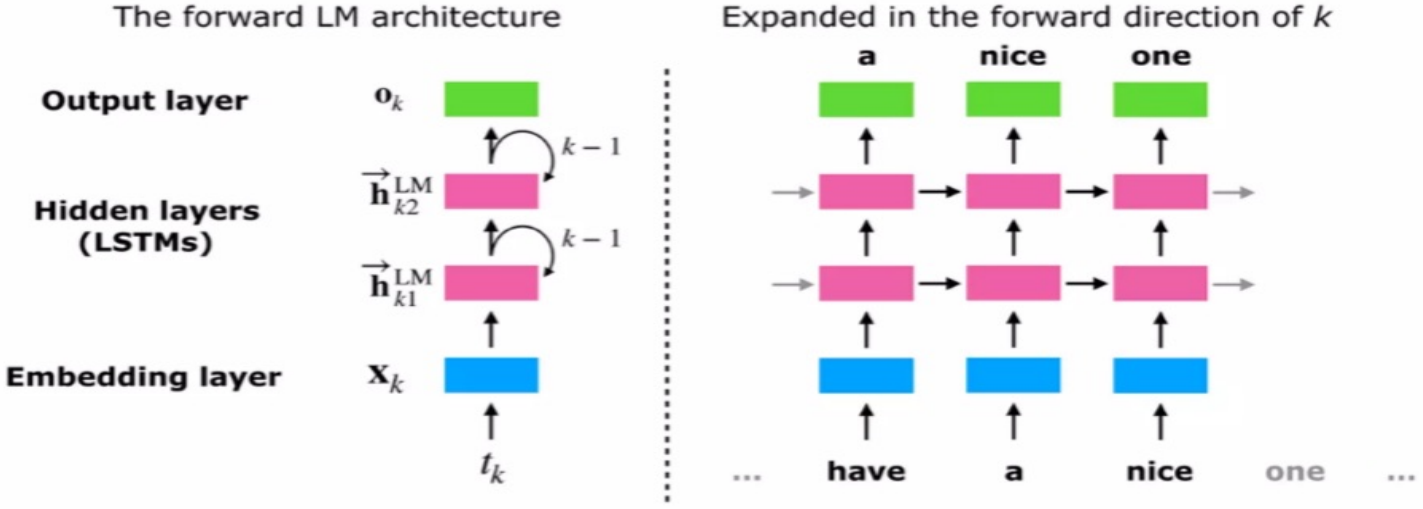
$$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}, \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_{k+1}, \ldots, t_N, \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right)$$

Θ_s → parameter for softmax layer.

35

Method

With long short term memory (LSTM) network, predicting the next words in both directions to build biLMs



36

ELMo represents a word t_k as a linear combination of corresponding hidden layers (inc. its embedding)

ELMo is a task specific representation. A down-stream task learns weighting parameters

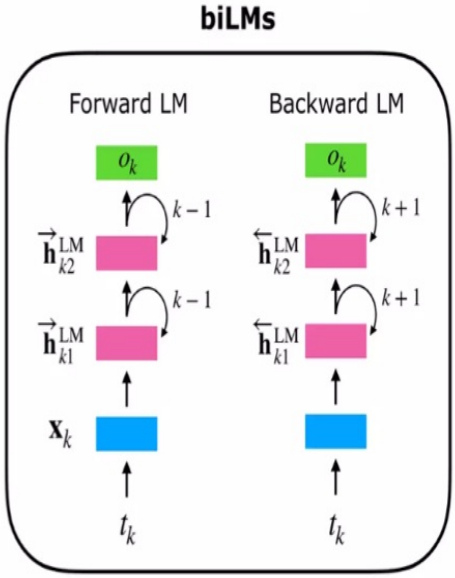
$$\mathbf{ELMo}_k^{task} = \gamma^{task} \times \sum \left\{ s_2^{task} \times \mathbf{h}_{k2}^{LM},\ s_1^{task} \times \mathbf{h}_{k1}^{LM},\ s_0^{task} \times \mathbf{h}_{k0}^{LM} \right.$$

$$\left. ([\mathbf{x}_k; \mathbf{x}_k]) \right.$$

Concatenate hidden layers

$$[\overrightarrow{\mathbf{h}}_{kj}^{LM}; \overleftarrow{\mathbf{h}}_{kj}^{LM}]$$

Unlike usual word embeddings, ELMo is assigned to every *token* instead of a *type*



37

ELMo is a task specific combination of the intermediate layer representations in the biLM. For each token t_k, a L-layer biLM computes a set of 2L+1 representations.

$$R_k = \{ x_k^{LM}, \overrightarrow{h}_{kj}^{LM}, \overleftarrow{h}_{kj}^{LM} \mid j=1 \dots L \}$$

$$= \{ h_{kj}^{LM} \mid j=0 \dots L \}$$

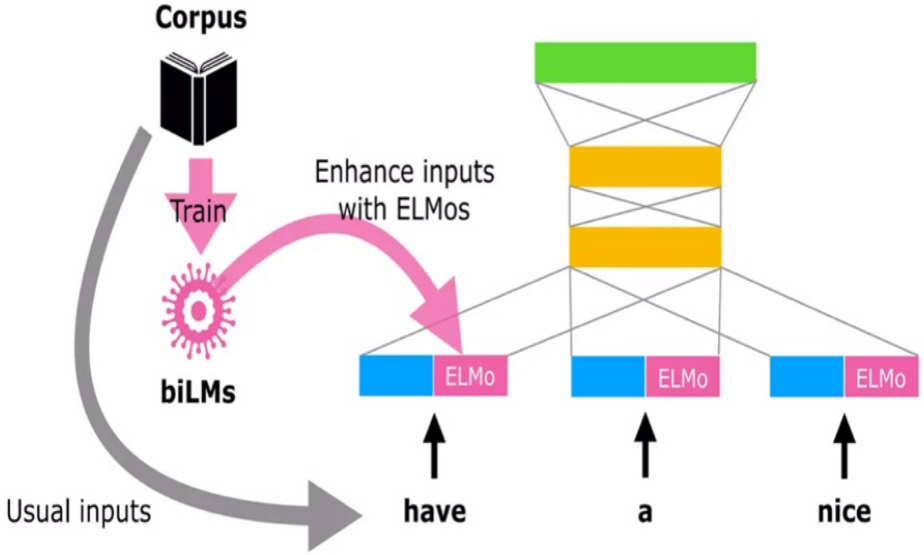
$$h_{jk}^{LM} = [\overrightarrow{h}_{kj}^{LM}, \overleftarrow{h}_{kj}^{LM}]$$ for each biLSTM layer.

Method

$$ELMo_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{kj}^{LM}$$

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer

- Overview
- Method
- Evaluation
- Analysis
- Comments



28

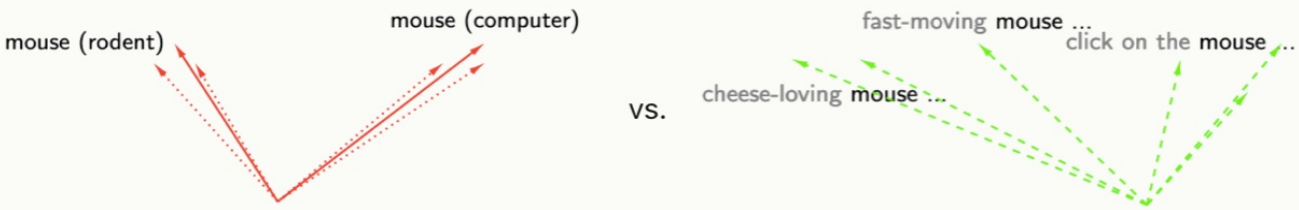
BERT, ELMo, & GPT-2: How contextual are contextualized word representations?

Feb 3, 2020 • Kawin Ethayarajh • (see paper)

Incorporating context into word embeddings - as exemplified by BERT, ELMo, and GPT-2 - has proven to be a watershed idea in NLP. Replacing *static vectors* (e.g., word2vec) with **contextualized word representations** has led to significant improvements on virtually every NLP task.

But just *how contextual* are these contextualized representations?

Consider the word 'mouse'. It has multiple word senses, one referring to a rodent and another to a device. Does BERT effectively create one representation of 'mouse' per word sense? Or does BERT create infinitely many representations of 'mouse', each highly specific to its context?



In our EMNLP 2019 paper, "How Contextual are Contextualized Word Representations?", we tackle these questions and arrive at some surprising conclusions:

1. In all layers of BERT, ELMo, and GPT-2, the representations of *all words* are anisotropic: they occupy a narrow cone in the embedding space instead of being distributed throughout.

2. In all three models, upper layers produce more context-specific representations than lower layers; however, the models contextualize words very differently from one another.

3. If a word's contextualized representations were not at all contextual, we'd expect 100% of their variance to be explained by a static embedding. Instead, we find that - on average - less than 5% of the variance can be explained by a static embedding.[1]

4. We can create a new type of static embedding for each word by taking the first principal component of its contextualized representations in a lower layer of BERT. Static embeddings created this way outperform GloVe and FastText on benchmarks like solving word analogies![2]

Going back to our example, this means that BERT creates highly context-specific representations of the word 'mouse' instead of creating one per word sense. Any static embedding of 'mouse' would account for very little of the variance in its contextualized representations. However, if we picked the vector that *did* maximize the variance explained, we would get a static embedding that is much better than the one provided by GloVe or FastText![3]

2-3. Measures of Contextuality

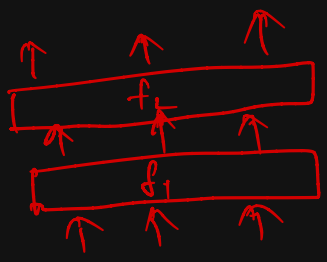
So, this paper explains contextuality through the following three indicators.

- self similarity: Similarity of the same word used in different sentences
- intra-sentence similarity: Similarity of words within one sentence
- maximum explainable variance: degree that can be expressed by static embedding

Let's look at each one one by one.

Self Similarity

$$\text{SelfSim}_l(\text{w}) = \frac{\text{One}}{n^2 - n} \sum_j \sum_{k \neq j} cos\left(f_l(s_j, i_j), f_l(s_k, i_k)\right)$$



- n: Number of sentences in which word w appears
- f: Models we use in experiments (ELMo, BERT, GPT)
- l: the corresponding layer of the model
- s_j: jth sentence
- i_j: Position where word w appears in sentence j
- $f_l(s_j, i_j)$: Output of the lth layer when using the f model of word w that appears in sentence s

Let's solve the above equation and talk about it.
Based on the embedding of BERT's 3rd layer,
it can be said to be the average of the similarity of word w to BERT's 3rd layer embedding in all sentences in which word w appears.

If word w is not contextualized at all, SelfSim = 1 because it will be the same embedding vector in all sentences.

Intra-Sentence Similarity

$$\text{IntraSim}_l(s) = \frac{\text{One}}{n} \sum_i cos\left(s_l, f_l(s, i)\right)$$

$$\text{where} \quad s_l = \frac{\text{One}}{n} \sum_i f_l(s, i)$$

- s_l: average of the embedding vectors for all tokens of sentence s in layer l of model f

To solve the above equation,
IntraSim is the average of the distance between the embedding vectors of each word in sentence l and the sentence vector in the corresponding model layer. This allows you to see how much a sentence is contextualized as it passes through the model, and how much the context affects the embedding vector for each word in that layer.

If both IntraSim and SelfSim are low, you can see that the corresponding layer of the model is embedding each word by reflecting its context, but each word is given a different context. SelfSim was lowered because a context was given, and IntraSim was lowered because a different context was given to each word.

If both IntraSim and SelfSim are low, you can see that the corresponding layer of the model is embedding each word by reflecting its context, but each word is given a different context. SelfSim was lowered because a context was given, and IntraSim was lowered because a different context was given to each word.

Conversely, in a situation where IntraSim is high but SelfSim is low, it means that all words are unified into one context. In the paper, it is expressed as “less nuanced contextualization,” and let’s explain it a little more (note, this is a brain official).

Being contextualized ultimately means processing a sentence as the sum of the following two elements.

Contextualization = original meaning of the word + meaning of the entire sentence
In this situation, contextualized embedding can be achieved only by retaining the unique meaning of each word to some extent (nuance) and incorporating the meaning of the entire sentence into each word. However, if IntraSim is high, each word loses its original information and only contains the information of the entire sentence.

BERT (By Google)

What is coreference resolution?

Coreference resolution (CR) is the task of finding all linguistic expressions (called mentions) in a given text that refer to the same real-world entity. After finding and grouping these mentions we can resolve them by replacing, as stated above, pronouns with noun phrases.

"I voted for Trump because he was most aligned with my values", John said.

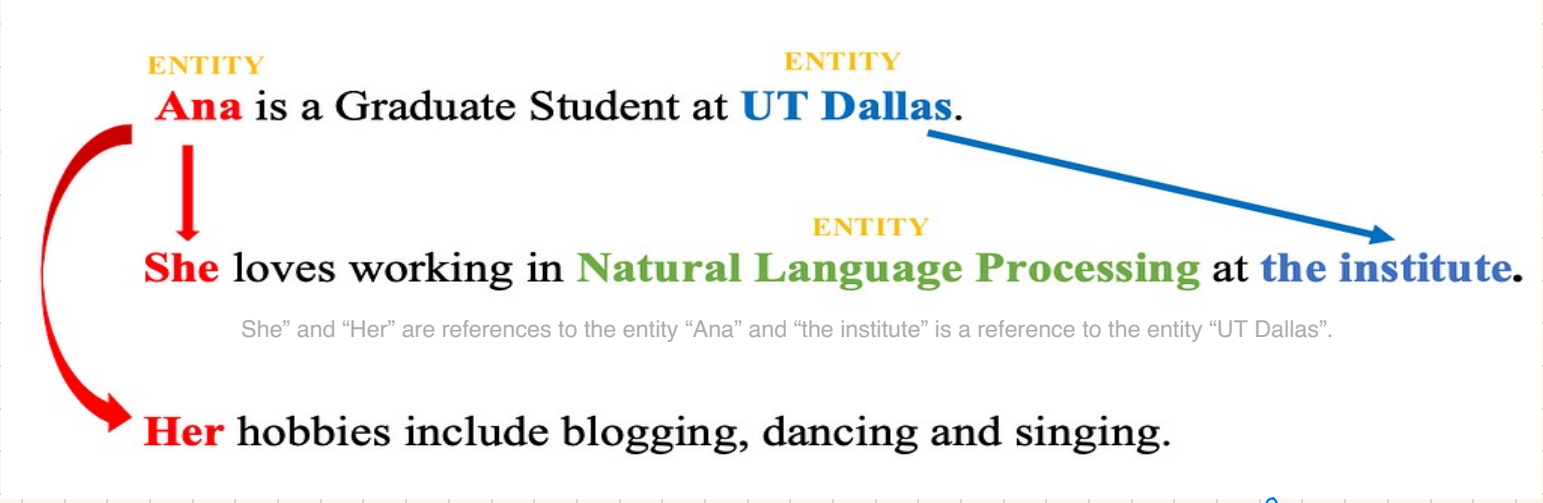
The original sentence

"John voted for Trump because Trump was most aligned with John's values", John said.

The sentence with resolved coreferences

Coreference resolution is an exceptionally versatile tool and can be applied to a variety of NLP tasks such as text understanding, information extraction, machine translation, sentiment analysis, or document summarization. It is a great way to obtain unambiguous sentences which can be much more easily understood by computers.

Discourse in the context of NLP refers to a sequence of sentences occurring one after the other. There will obviously be entities that are being talked about and possible references to those entities in the discourse. We use the word "mention" to refer to these references.



She" and "Her" are references to the entity "Ana" and "the institute" is a reference to the entity "UT Dallas".

Here, "Ana", "Natural Language processing" & "UT Dallas" are possible entities.

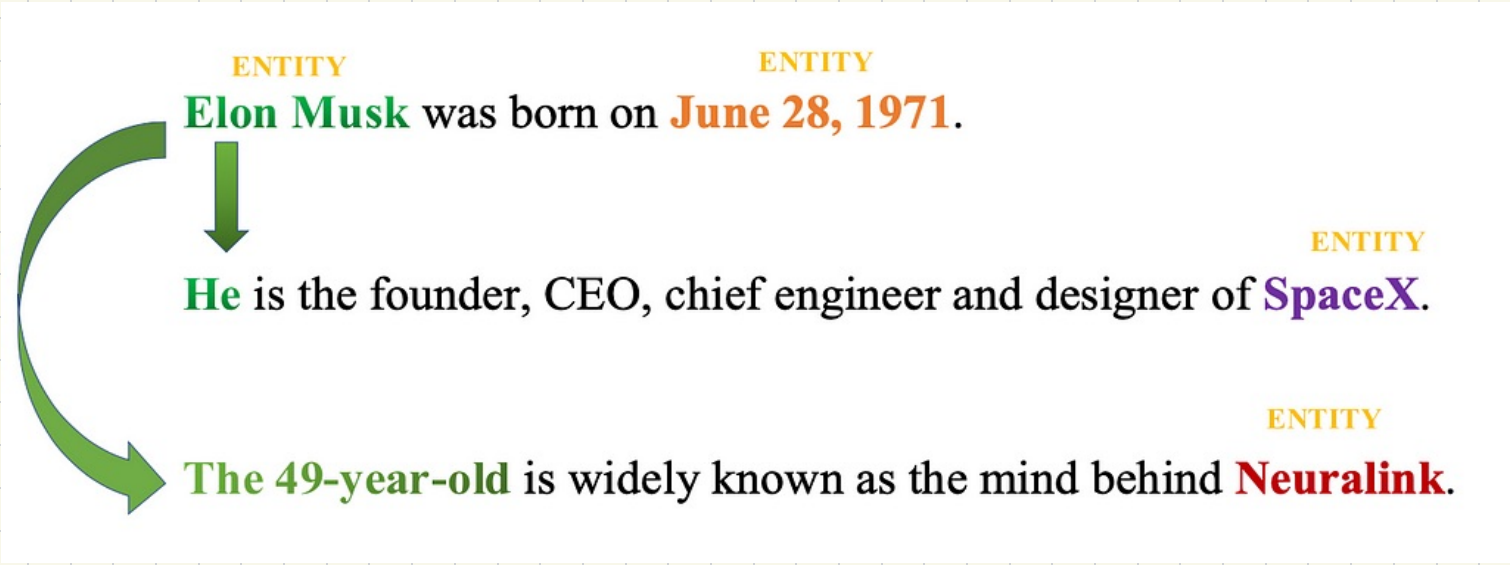
"She" & "Her" are references to the entity "Ana" and "the institute" is a reference to the entity "UT Dallas"

***Reference**, in NLP, is a linguistic process where one word in a sentence or discourse may refer to another word or entity. The task of resolving such references is known as **Reference Resolution**. In the above example, "She" and "Her" referring to the entity "Ana" and "the institute" referring to the entity "UT Dallas" are two examples of Reference Resolution.*

Coreference Resolution in particular, is the process of resolving pronouns to identify which entities are they referring to. It is also a kind of Reference Resolution. The entities resolved may be a person, place, organization, or event.

Referent is the object that is being referred to. For example, "Ana" is the referent in the above example.
Referring expression are the mentions or linguistic expressions given in the discourse.
Two or more referring expressions that refer to the same discourse entity are said to corefer.

"Elon Musk was born on June 28, 1971. He is the founder, CEO , chief engineer and designer of SpaceX. The 49 year old is widely known as the mind behind Neuralink."



Referring Expressions: Elon Musk, He, The 49 year old

Referent: Elon Musk

Corefering Expressions: {Elon Musk, He}, {Elon Musk, The 49 year old}

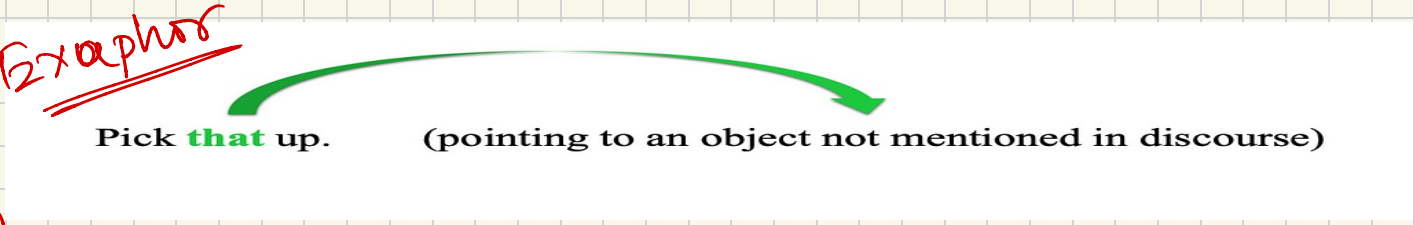
References are usually of two kinds: Exaphor and Endophor. Endophor refers to an entity that appears in the discourse. While Exaphor refers to an entity that does not appear in the discourse.

References
Exaphor (doesn't appear in the discourse)
Endophor (within the discourse)

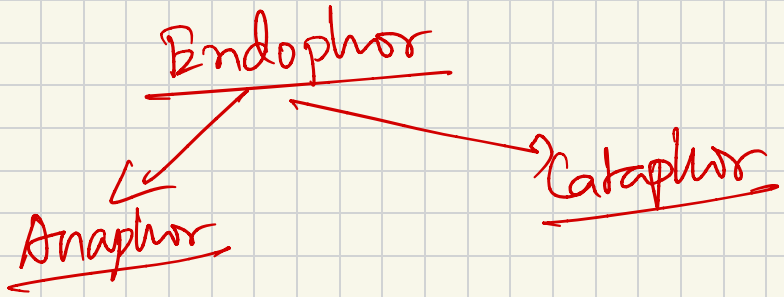
Endophor

Here “She” refers to “Ana” which appears as a possible referrent that is mentioned explicitly in the discourse.

Exophor



Here “that” refers to a object which appears as a possible referrent for a object that it not mentioned explicitly in the discourse



There are primarily two kinds of Endophors: Anaphor and Cataphor. Anaphor refers to a situation wherein the referential entity or referent appears before its referencing pronoun in the discourse.

Here “She” refers to “Ana” whose occurence precedes the occurence of its referencing pronoun “She” in the discourse.

Antecedent – The entity to which the anaphor refers.

While, Cataphor refers to a situation wherein the entity or referent occurs later than its referencing pronoun in the discourse.

Here “she” occurs before its referential entity or referent “Ana” in the discourse. Thus, this is an example of cataphor.

Some of the linguistic properties which we will be talking about are:

Number and Gender Agreement

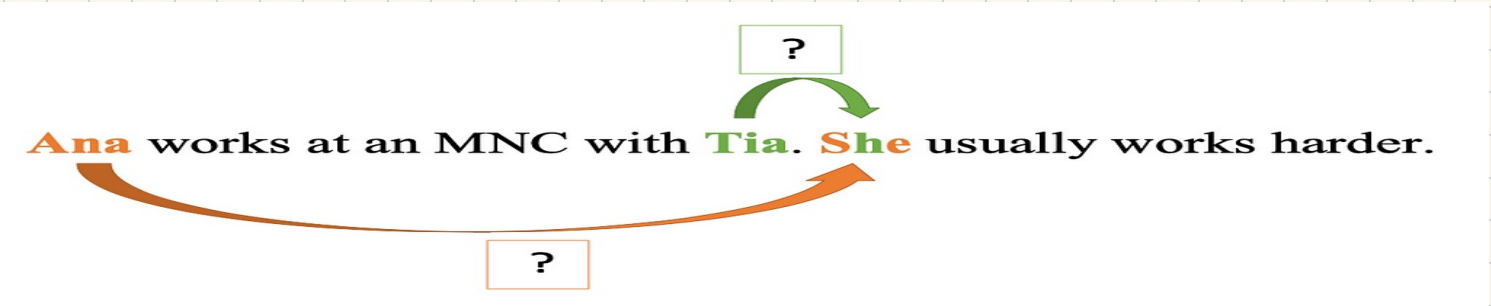
Number Agreement basically means that the referencing expressions should agree in number. While Gender Agreement implies that the referencing expressions agree in gender.

Here, “Analisa” and “She” agree in gender Female and they agree in number i.e., only one person “Analisa” works at Google and thus we use “She” to reference her rather than using other pronouns like “he”, “they”, etc. This is gender and number agreement.

Here, “horses” are plural. Hence, the referent used is “they” in order to refer to the entity “horses”.Thus, they agree on numbers. This is number agreement.

Grammatical Role

Another property to remember is the Grammatical role. This property takes advantage of the inherent grammatical nature of a sentence which gives more saliency value to subject entity as compared to an object entity. In other words, we assume that an entity which is a subject is usually more important than an object entity.



In this sentence, we have “Ana” and “Tia” as candidate referents for the word “She”. Here “Ana” is the subject while “Tia” is the object. Thus, keeping in mind the saliency, we deem “Ana” to be coreferent to “She” rather than “Tia” .

Verb Semantics

Some verbs tend to lend more meaning to one of their arguments as compared to others while performing semantic analysis.

"Ana helped Christa. She was the architect behind the project."

Here, in the first sentence, the usage of verb "helped" implies that the probability of Ana being the architect behind the project is higher than Christa. Thus, "She" refers to "Ana" in the first sentence.

"Ana condemned Christa. She was the architect behind the project."

However, in the second sentence, the usage of verb "condemned" implies that the probability of Christa being the architect behind the project is higher than Ana. Thus, "She" refers to "Christa" in the second sentence.

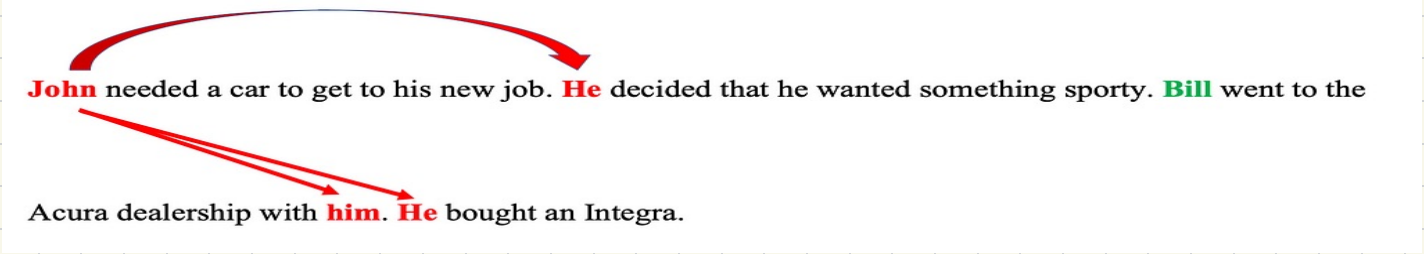
Selectional Restrictions

Here, two possible referents for "it" are "pajamas" and "chicken". The usage of the verb "eat" ("ate" is past tense for "eat") implies that the referent entity must be edible, thereby choosing "chicken" as the referent for "it".

Repeated Mention

Another essential feature to understand is Repeated Mention. This feature or property says that if an entity or a set of entities are referred to repeatedly in the discourse, then the probability of them being possible referents increases exponentially.

"John needed a car to get to his new job. He decided that he wanted something sporty. Bill went to the Acura dealership with him. He bought an Integra."



Here, the repeated mention of "John" as compared to "Bill" as the focal point implies that "He" refers to "John" and not "Bill".

Parallelism

This property lends more importance to a referent if it can draw similar properties in terms of syntactic and semantic information from another sentence.

Mary went with Sue to the Acura dealership.
Sally went with her to the Mazda dealership.

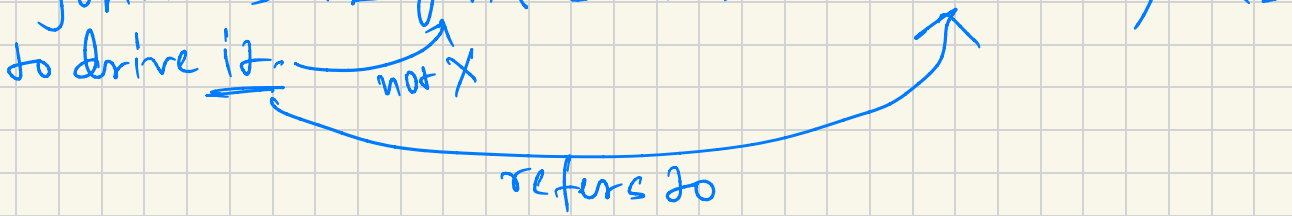
Here, “her” refers to Sue as both the sentences imply similar synatactic and semantic structure and we can draw parallels between them.

Recency

Entities introduced recently are more salient than those introduced before.

Ex:

John has a Legend. Bill has an Escort. Mary likes to drive it.



Winograd schema challenge:

Contains:

- A pair of sentences that differ in a single word or phrase.
- A coreference question.

Ex:

Sentence: The trophy didn't fit in the suitcase because it was too large.

Question: what was too large? Ans: The trophy

Sentence: The trophy didn't fit into the suitcase because it was too small.

Question: What was too small? Ans: The suitcase.

Discourse:

refers to a sequence of sequences occurring one after another.

two steps to solve Coreference Resolution:

1. Detect the mentions (easy)

"[I] voted for [Nader] because [he] was most aligned with [[my] values]" she said

mention can be nested.

2. cluster the mentions (hard)

"[I] voted for (Nader) because (he) was most aligned with [[my] values]" she said

Mention detection:

Mention: span of text referring to some entity.

Three kinds of mentions:

1. Pronouns: I, your, it, she, him etc.
2. Named Entity: people, places etc.
3. Noun phrases: "a dog", "the big fluffy cat"

Is mention detection easy task?

NO, are these mentions?

- It is sunny

- Every student
- No student
- the best donut in the world.
- 100 miles.

How to deal with these bad mentions?

→ Could train a classifier to filter out spurious mentions.

→ Much more common: Keep all mentions as "candidate mentions" → after clustering, discard all singleton mentions.

** The mention-Pair Architecture:

Given:
→ pair of mentions (candidate anaphor & candidate antecedent).

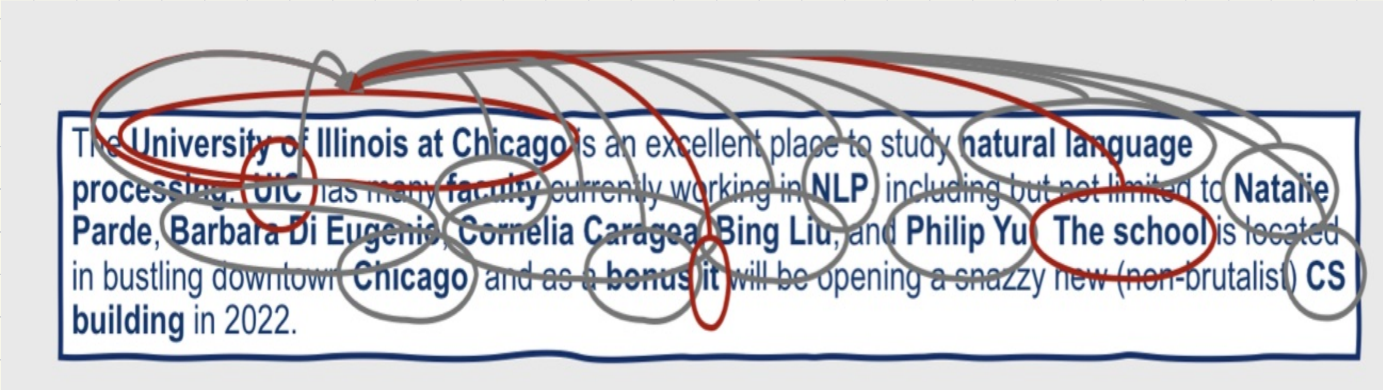
Decide:
whether or not they corefer.

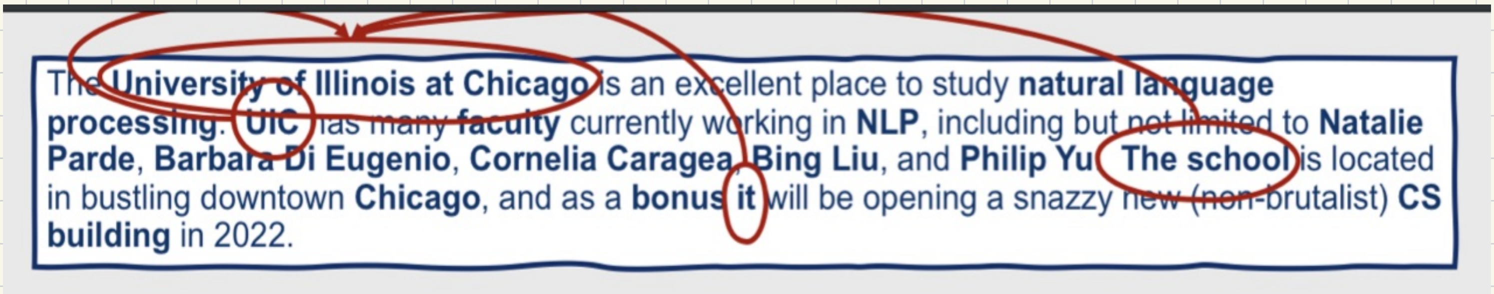
How does it work?

→ Compute coreference probabilities for every plausible pair of mentions.

→ Goal: High probability of actual coreferring pairs, & low probability for other pairs.

The **University of Illinois at Chicago** is an excellent place to study **natural language processing**. **UIC** has many **faculty** currently working in **NLP**, including but not limited to **Natalie Parde**, **Barbara Di Eugenio**, **Cornelia Caragea**, **Bing Liu**, and **Philip Yu**. The **school** is located in bustling downtown **Chicago** and as a **bonus** it will be opening a snazzy new (non-brutalist) **CS building** in 2022.





How do we learn these probabilities?

- Select training samples
  - One positive instance (m_i, m_j) where m_j is the closest antecedent to m_i
  - A negative instance (m_i, m_k) for each m_k between m_j and m_i
- Extract features
  - Hand-built features, and/or
  - Implicitly learned representations
- Train classification model

How do we make predictions?

- Apply the trained classifier to each test instance in a clustering step
  - **Closest-first clustering**
    - For mention *i*, classifier is run backwards through prior *i*-1 mentions
    - First antecedent with probability > 0.5 is selected and linked to *i*
  - **Best-first clustering**
    - Classifier is run on all possible *i*-1 antecedents
    - Mention with highest probability is selected as the antecedent for *i*

Advantage: Simplest coreference resolution architecture.

Disadvantage: → Doesn't directly compare candidate antecedents with one another.

→ Considers only mentions, not overall entities.

P(coref | "victoria chen", "she")

victoria chen Megabucks Banking her her pay the 37 year old she

P(coref | "Megabucks Banking", "she")

for a pair of mentions, predicts wheather it is a coreference.

How to address these limitations? :—

22.4.2 The Mention-Rank Architecture

The mention ranking model directly compares candidate antecedents to each other, choosing the highest-scoring antecedent for each anaphor.

In early formulations, for mention i, the classifier decide which of the $\{1, \ldots, i-1\}$ prior mentions is the antecedent (Denis and Baldridge, 2008). But suppose i is in fact not anaphoric, and none of the antecedents should be chosen? Such a model would need to run a separate anaphoricity classifier on i. Instead, it turns out to be better to jointly learn anaphoricity detection and coreference together with a single loss (Rahman and Ng, 2009).

So in modern mention-ranking systems, for the ith mention (anaphor), we have an associated random variable y_i ranging over the values $Y(i) = \{1, \ldots, i-1, \varepsilon\}$. The value ε is a special dummy mention meaning that i does not have an antecedent (i.e., is either discourse-new and starts a new coref chain, or is non-anaphoric).

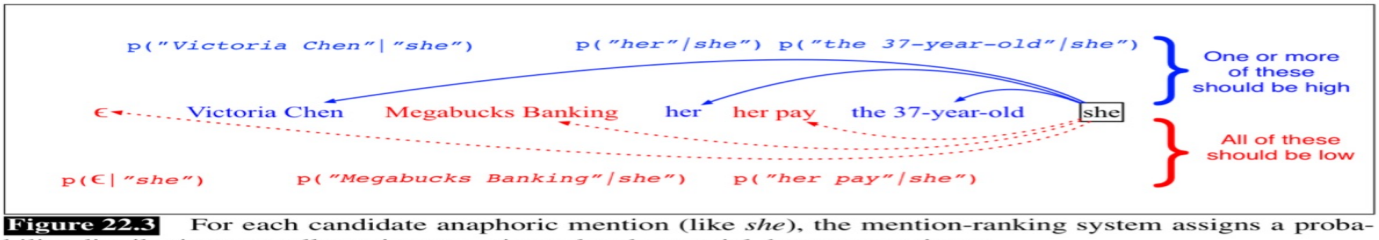


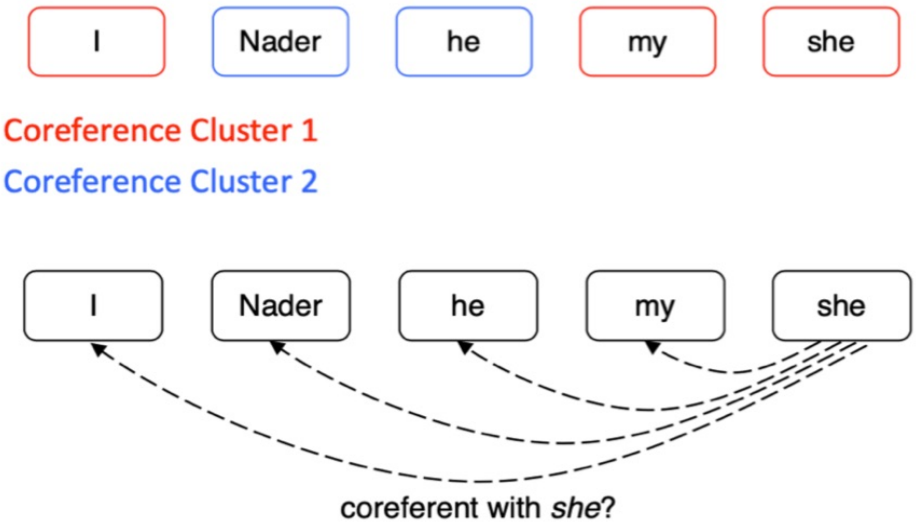
Figure 22.3 For each candidate anaphoric mention (like *she*), the mention-ranking system assigns a probability distribution over all previous mentions plus the special dummy mention ε.

At test time, for a given mention i the model computes one softmax over all the antecedents (plus ε) giving a probability for each candidate antecedent (or none).

Mention-pair models

- Train a binary classifier that assigns every pair of mentions a probability of being coreferent: $p(m_i, m_j)$

*"**I** voted for **Nader** because **he** was most aligned with **my** values," **she** said.*

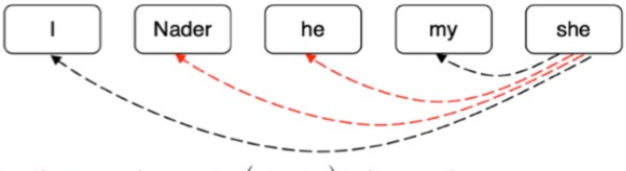


Mention-pair models

*"**I** voted for **Nader** because **he** was most aligned with **my** values," **she** said.*

Positive examples: want $p(m_i, m_j)$ to be near 1

*"**I** voted for **Nader** because **he** was most aligned with **my** values," **she** said.*



Negative examples: want $p(m_i, m_j)$ to be near 0

$$J = -\sum_{i=2}^{N}\sum_{j=1}^{i} y_{ij} \log p(m_j, m_i)$$

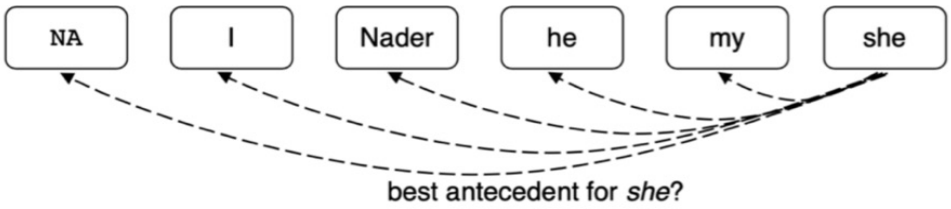
Iterate through mentions

Iterate through candidate antecedents (previously occurring mentions)

Coreferent mentions pairs should get high probability, others should get low probability

Mention-ranking models

- Assign each mention its highest scoring candidate antecedent according to the model
- Add a dummy NA mention to decline linking the current mention to anything ("singleton" or "first" mention)



p(NA, she) = 0.1
p(I, she) = 0.5
p(Nader, she) = 0.1
p(he, she) = 0.1
p(my, she) = 0.2

Apply a softmax over the scores for candidate antecedents so probabilities sum to 1

Mention-ranking models

- Training time: only clustering information is observed (no annotation of "antecedent"), so we optimize the marginal log-likelihood of all the correct antecedents.

$$J = \sum_{i=2}^{N} -\log \left(\sum_{j=1}^{i-1} \mathbb{1}(y_{ij} = 1) p(m_j, m_i) \right)$$

Iterate over all the mentions in the document

Usual trick of taking negative log to go from likelihood to loss

- Testing time: same as mention-pair but we only pick one antecedent for each mention

End-to-End Neural Coreference Resolution :

End-to-end coreference resolution

- A mention-ranking model
- Joint mention detection and clustering — so you don't need an additional mention detector (parser/part-of-speech tagger)

$$J = \sum_{i=2}^{N} -\log \left(\sum_{j=1}^{i-1} \mathbb{1}(y_{ij} = 1) p(m_j, m_i) \right)$$

Iterate over all the mentions in the document

Usual trick of taking negative log to go from likelihood to loss

We consider all the possible spans + {NA}

$$p(m_j, m_i) = \frac{\exp(s(m_j, m_i))}{\sum_{j' < i} \exp(m_{j'}, m_i)}$$

$$N = \frac{T(T+1)}{2} + 1$$

T: number of words

(Lee et al, 2017): End-to-end Neural Coreference Resolution

End-to-end coreference resolution

$$s(i,j) = s_{\mathrm{m}}(i) + s_{\mathrm{m}}(j) + s_{\mathrm{a}}(i,j)$$

Are spans *i* and *j* coreferent mentions?

Is *i* a mention?

Is *j* a mention?

Do they look coreferent?

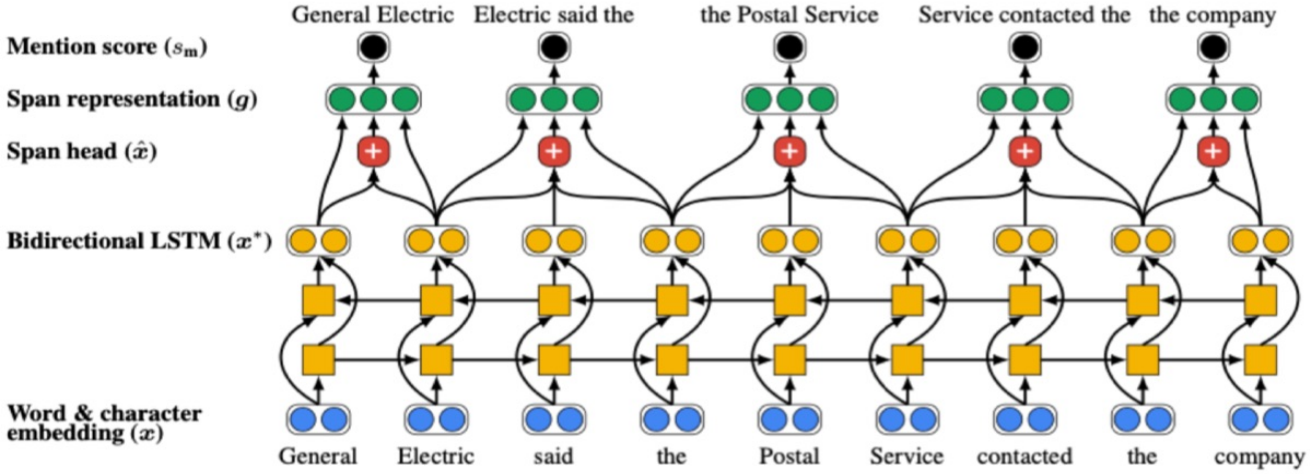
$$s_{\mathrm{m}}(i) = \boldsymbol{w}_{\mathrm{m}} \cdot \mathrm{FFNN}_{\mathrm{m}}(\boldsymbol{g}_i)$$

$$s_{\mathrm{a}}(i,j) = \boldsymbol{w}_{\mathrm{a}} \cdot \mathrm{FFNN}_{\mathrm{a}}([\boldsymbol{g}_i, \boldsymbol{g}_j, \boldsymbol{g}_i \circ \boldsymbol{g}_j, \phi(i,j)])$$

Let's compute a vector representation $\mathbf{g}_i \in \mathbb{R}^d$ for each span i

$\phi(i,j)$: manual features such speaker/gender information

End-to-end coreference resolution



Span representation: $\boldsymbol{g}_i = [\boldsymbol{x}^*_{\text{START}(i)}, \boldsymbol{x}^*_{\text{END}(i)}, \hat{\boldsymbol{x}}_i, \phi(i)]$

BILSTM hidden states for span's start and end

Attention-based representation (details next slide) of the words in the span

Additional features

(Lee et al, 2017): End-to-end Neural Coreference Resolution

End-to-end coreference resolution

Attention scores

$$\alpha_t = \boldsymbol{w}_\alpha \cdot \text{FFNN}_\alpha(\boldsymbol{x}^*_t)$$

dot product of weight vector and transformed hidden state

Attention distribution

$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)}$$

just a softmax over attention scores for the span

Final representation

$$\hat{\boldsymbol{x}}_i = \sum_{t=\text{START}(i)}^{\text{END}(i)} a_{i,t} \cdot \boldsymbol{x}_t$$

Attention-weighted sum of word embeddings

$$\boldsymbol{g}_i = [\boldsymbol{x}^*_{\text{START}(i)}, \boldsymbol{x}^*_{\text{END}(i)}, \hat{\boldsymbol{x}}_i, \phi(i)]$$

hidden states for span's start and end

Attention-based representation

Additional features

Represents the context to the left and right of the span

Represents the span itself

Represents other information not in the text

Parsing

→ The construction of the sentence is governed by grammatical rules.

→ How the words connect with each other according to the rule - structural composition of the sentence.

→ Identifying the structural composition — parsing.

1. Constituency parsing
2. Dependency parsing
3. Semantic parsing.

Constituency parsing is a natural language processing technique that is used to analyze the grammatical structure of sentences. It is a type of syntactic parsing that aims to identify the constituents, or subparts, of a sentence and the relationships between them. The output of a constituency parser is typically a parse tree, which represents the hierarchical structure of the sentence.

The process of constituency parsing involves identifying the syntactic structure of a sentence by analyzing its words and phrases. This typically involves identifying the noun phrases, verb phrases, and other constituents, and then determining the relationships between them. The parser uses a set of grammatical rules and a grammar model to analyze the sentence and construct a parse tree.

17.2 Context-Free Grammars

CFG

A widely used formal system for modeling constituent structure in natural language is the **context-free grammar**, or **CFG**. Context-free grammars are also called **phrase-structure grammars**, and the formalism is equivalent to **Backus-Naur form**, or **BNF**. The idea of basing a grammar on constituent structure dates back to the psychologist Wilhelm Wundt (1900) but was not formalized until Chomsky (1956) and, independently, Backus (1959).

rules

A context-free grammar consists of a set of **rules** or **productions**, each of which expresses the ways that symbols of the language can be grouped and ordered together, and a **lexicon** of words and symbols. For example, the following productions express that an **NP** (or **noun phrase**) can be composed of either a *ProperNoun* or a determiner (*Det*) followed by a *Nominal*; a *Nominal* in turn can consist of one or

lexicon

NP

more *Nouns*.[1]

$$\begin{aligned} NP &\rightarrow Det\ Nominal \\ NP &\rightarrow ProperNoun \\ Nominal &\rightarrow Noun \mid Nominal\ Noun \end{aligned}$$

Context-free rules can be hierarchically embedded, so we can combine the previous rules with others, like the following, that express facts about the lexicon:

$$\begin{aligned} Det &\rightarrow a \\ Det &\rightarrow the \\ Noun &\rightarrow flight \end{aligned}$$

terminal

non-terminal

The symbols that are used in a CFG are divided into two classes. The symbols that correspond to words in the language ("the", "nightclub") are called **terminal** symbols; the lexicon is the set of rules that introduce these terminal symbols. The symbols that express abstractions over these terminals are called **non-terminals**. In each context-free rule, the item to the right of the arrow ($\rightarrow$) is an ordered list of one or more terminals and non-terminals; to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. The non-terminal associated with each word in the lexicon is its lexical category, or part of speech.

A CFG can be thought of in two ways: as a device for generating sentences and as a device for assigning a structure to a given sentence. Viewing a CFG as a generator, we can read the $\rightarrow$ arrow as "rewrite the symbol on the left with the string of symbols on the right".

So starting from the symbol:	*NP*
we can use our first rule to rewrite *NP* as:	*Det Nominal*
and then rewrite *Nominal* as:	*Noun*
and finally rewrite these parts-of-speech as:	*a flight*

We say the string *a flight* can be derived from the non-terminal *NP*. Thus, a CFG can be used to generate a set of strings. This sequence of rule expansions is called a derivation of the string of words. It is common to represent a derivation by a **parse tree** (commonly shown inverted with the root at the top). Figure 17.1 shows the tree representation of this derivation.

derivation

parse tree

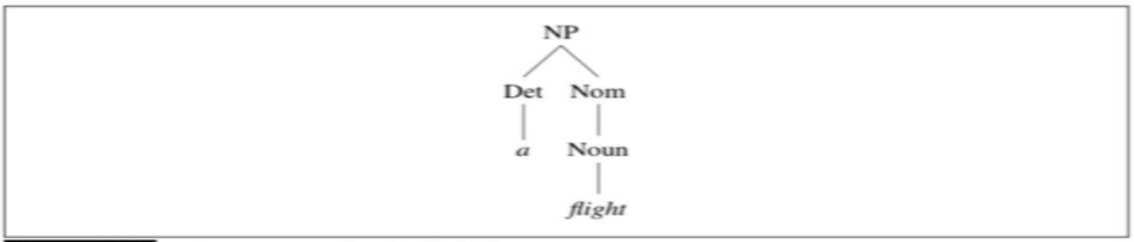


Figure 17.1 A parse tree for "a flight".

17.2.1 Formal Definition of Context-Free Grammar

We conclude this section with a quick, formal description of a context-free grammar and the language it generates. A context-free grammar G is defined by four parameters: N, Σ, R, S (technically it is a "4-tuple").

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)^*$
S	a designated **start symbol** and a member of N

Grammar Rules			Examples
S	$\rightarrow$	*NP VP*	I + want a morning flight
NP	$\rightarrow$	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	$\rightarrow$	*Preposition NP*	from + Los Angeles

Figure 17.3 The grammar for $\mathcal{L}_0$, with example phrases for each rule.

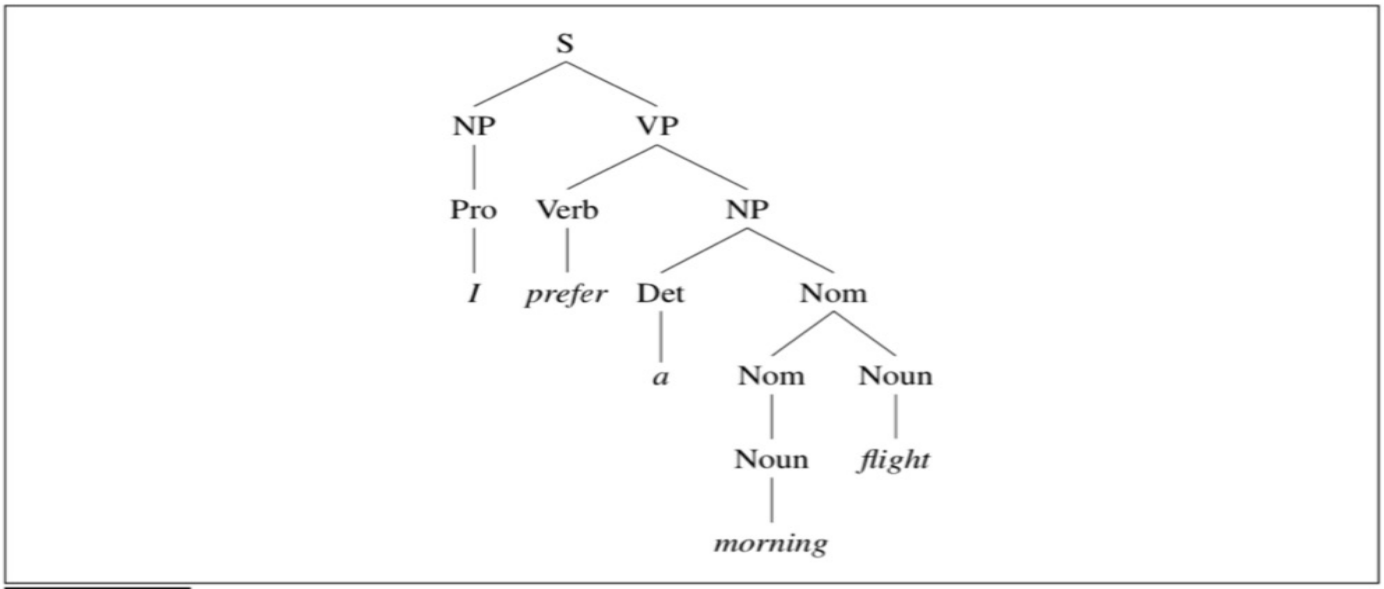


Figure 17.4 The parse tree for "I prefer a morning flight" according to grammar $\mathcal{L}_0$.

Treebanks

A corpus in which every sentence is annotated with a parse tree is called a treebank.

Treebanks play an important role in parsing as well as in linguistic investigations of syntactic phenomena.

Treebanks are generally made by parsing each sentence with a parse that is then hand-corrected by human linguists. Figure 17.5 shows sentences from the **Penn Treebank** project, which includes various treebanks in English, Arabic, and Chinese. The Penn Treebank part-of-speech tagset was defined in Chapter 8, but we'll see minor formatting differences across treebanks. The use of LISP-style parenthesized notation for trees is extremely common and resembles the bracketed notation we saw earlier in (17.1). For those who are not familiar with it we show a standard node-and-line tree representation in Fig. 17.6.

```
((S
   (NP-SBJ (DT That)
     (JJ cold) (, ,)
     (JJ empty) (NN sky) )
   (VP (VBD was)
     (ADJP-PRD (JJ full)
       (PP (IN of)
         (NP (NN fire)
           (CC and)
           (NN light) ))))
   (. .) ))
```

(a)

```
((S
   (NP-SBJ The/DT flight/NN )
   (VP should/MD
     (VP arrive/VB
       (PP-TMP at/IN
         (NP eleven/CD a.m/RB ))
       (NP-TMP tomorrow/NN )))))
```

(b)

Figure 17.5 Parses from the LDC Treebank3 for (a) Brown and (b) ATIS sentences.

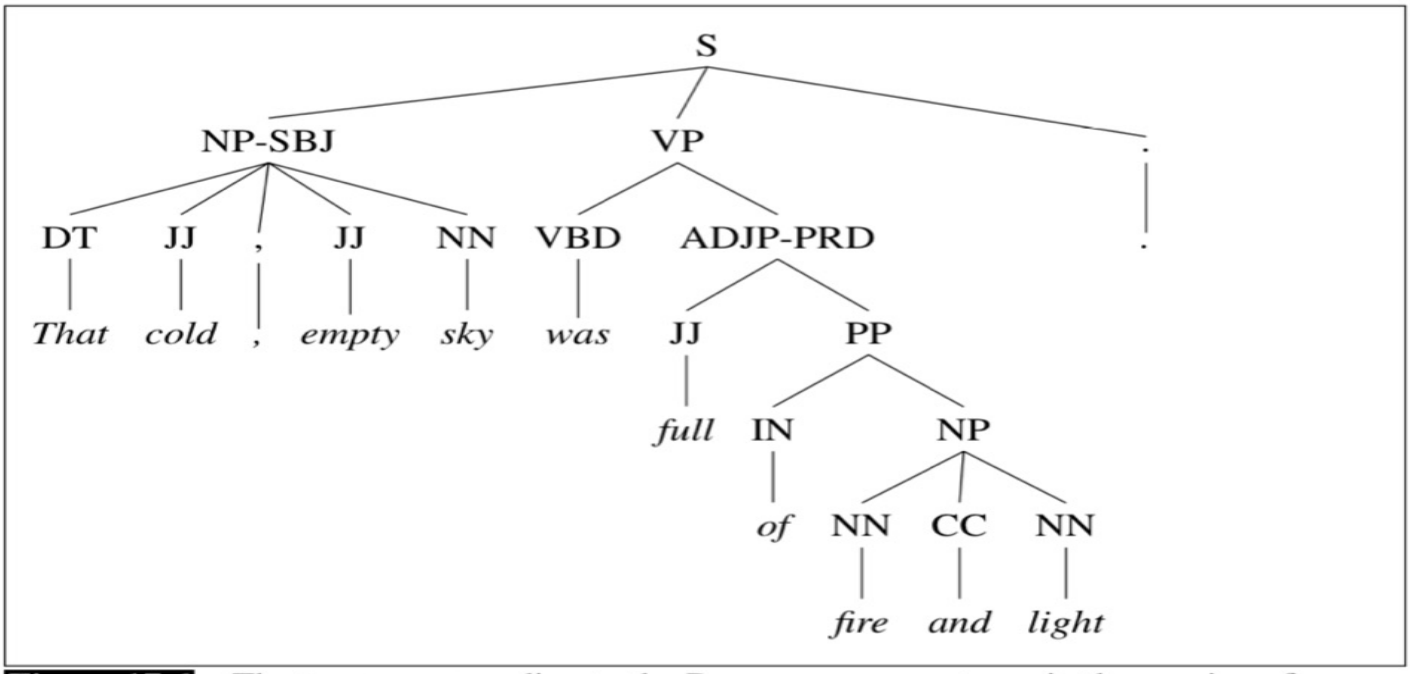


Figure 17.6 The tree corresponding to the Brown corpus sentence in the previous figure.

The sentences in a treebank implicitly constitute a grammar of the language. For example, from the parsed sentences in Fig. 17.5 we can extract the CFG rules shown in Fig. 17.7 (with rule suffixes (-SBJ) stripped for simplicity). The grammar used to parse the Penn Treebank is very flat, resulting in very many rules. For example,

17.5 Ambiguity

structural ambiguity

Ambiguity is the most serious problem faced by syntactic parsers. Chapter 8 introduced the notions of **part-of-speech ambiguity** and **part-of-speech disambiguation**. Here, we introduce a new kind of ambiguity, called **structural ambiguity**, illustrated with a new toy grammar $\mathcal{L}_1$, shown in Figure 17.8, which adds a few rules to the $\mathcal{L}_0$ grammar from the last chapter.

Structural ambiguity occurs when the grammar can assign more than one parse to a sentence. Groucho Marx's well-known line as Captain Spaulding in *Animal*

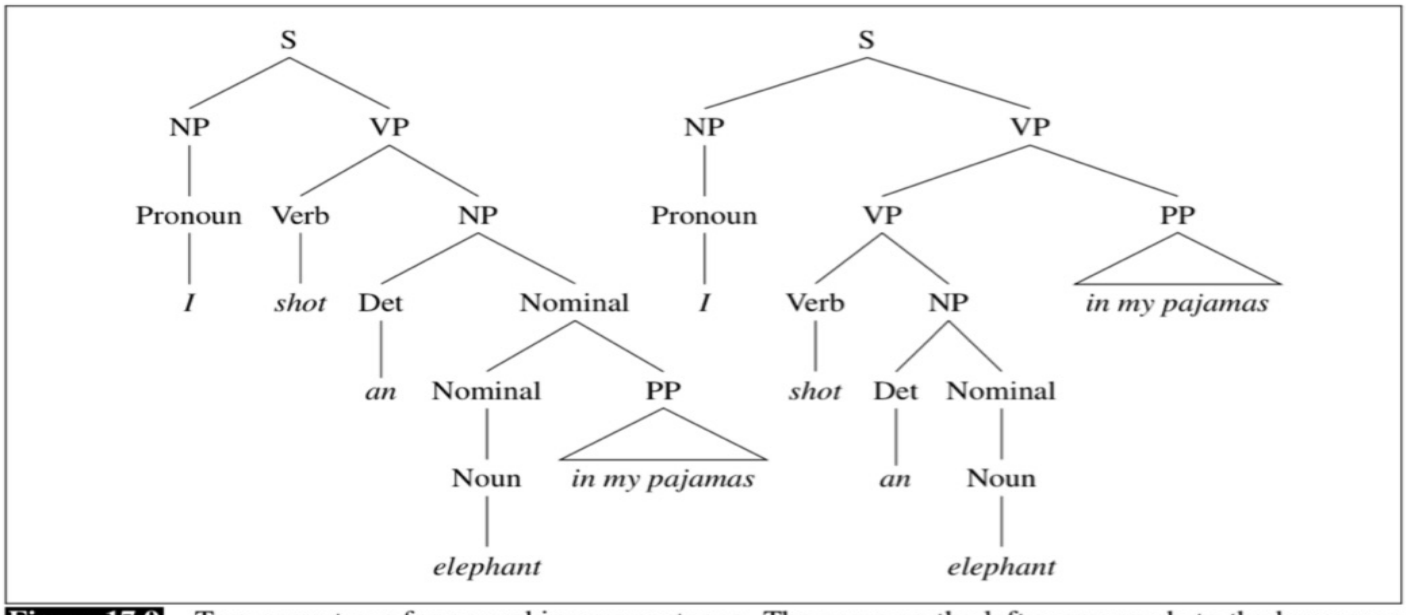


Figure 17.9 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

Crackers is ambiguous because the phrase *in my pajamas* can be part of the *NP* headed by *elephant* or a part of the verb phrase headed by *shot*. Figure 17.9 illustrates these two analyses of Marx's line using rules from $\mathcal{L}_1$.

CKY Parsing: A Dynamic Programming Approach

Dynamic programming provides a powerful framework for addressing the prob- lems caused by ambiguity in grammars.

- Requires the grammer to be in chomsky Normal Form (CNF).
  - → restricted to rules of the form
    $A \rightarrow BC$
    or
    $A \rightarrow w \mid \phi$

steps of cyk algorithm:

1. Convert CFG to CNF
2. Construct a triangular table.
3. check if $x_{1,n}$ contains start symbols.

time complexity of cyk $\Rightarrow O(n^3)$
where n is the no of words.

function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$
  for $i \leftarrow$ **from** $j-2$ **down to** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

Figure 17.12 The CKY algorithm.

Let's consider CNF below.

S → NP VP, 0.4
PP → IN NP, 0.1
NP → DET NP, 0.3
NP → NP PP, 0.1
VP → VBD NP, 0.2
VP → VP PP, 0.3
NP → PRP$ NP, 0.5

DET → "an", 0.9
VBD → "shot", 0.3
NP → "pajamas", 0.8
NP → "elephant", 0.9
NP → "I", 0.2
PRP → "I", 0.6
IN → "in", 0.9
PRP$ → "my" 0.8

I shot an elephant in my pajamas

NP/PRP	ϕ	ϕ	S	ϕ	ϕ	S1/S2/S3
	VBD	ϕ	VP	ϕ	ϕ	VP1/VP2/VP3
		DET	NP	ϕ	ϕ	NP1/NP2
			NP	ϕ	ϕ	NP
				IN	ϕ	PP
					PRP$	NP
						NP

no rules producing NP VBD or PRP VBD so keep it ϕ

"an" "elephant"
NP → (DET NP)

"an" "elephant in my pajamas"
(DET NP) → NP

"an elephant" "in my pajamas"
(NP PP) → NP

three valid parsers.
But how to find the best one?
↓
use PCFG (probabilistic CFG)
same as CFG except each rule
A → β in the grammar is associated with a probability P(β|A).