Natural Language Processing (NLP)
State of the art models - Technical class

Joan Ginés i Ametllé

AI Researcher

23rd July 2020

cinnamon AI

Outline

1. Introduction to language technologies
2. How to turn words into features
3. How to translate between languages
4. How to classify texts
5. Transformers for natural language
6. Conclusion

1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

A little bit about languages

Language

A capacity for communication that humans possess and that distinguishes them from other animals.

A language

A specific instantiation of *Language* (e.g. Vietnamese, English). There are around 6500 in use today.

A little bit about languages

Language

A capacity for communication that humans possess and that distinguishes them from other animals.

A language

A specific instantiation of *Language* (e.g. Vietnamese, English). There are around 6500 in use today.

A little bit about languages

Language

A capacity for communication that humans possess and that distinguishes them from other animals.

A language

A specific instantiation of *Language* (e.g. Vietnamese, English). There are around 6500 in use today.

A little bit about languages

Language

A capacity for communication that humans possess and that distinguishes them from other animals.

A language

A specific instantiation of *Language* (e.g. Vietnamese, English). There are around 6500 in use today.

English	German	Chinese	Japanese
welcome	willkommen	欢迎	歓迎
Hindi	French	Korean	Arabic
आपका स्वागत है	bienvenue	환영	نرحب

A little bit about languages

Some distinctive features of language:

- **Arbitrariness**: There is no inherent relationship between form and meaning (e.g. *horse*, *pferd*, *cavalo*).
- **Duality**: A finite number of phonemes can combine to form an infinite number of words.
- **Discreteness**: The sound elements are perceived categorically, despite actually being continuous.
- **Productivity**: We can produce and understand a nearly infinite number of messages.

A little bit about languages

Some distinctive features of language:

- **Arbitrariness**: There is no inherent relationship between form and meaning (e.g. *horse*, *pferd*, *cavalo*).
- **Duality**: A finite number of phonemes can combine to form an infinite number of words.
- **Discreteness**: The sound elements are perceived categorically, despite actually being continuous.
- **Productivity**: We can produce and understand a nearly infinite number of messages.

A little bit about languages

Some distinctive features of language:

- **Arbitrariness**: There is no inherent relationship between form and meaning (e.g. *horse*, *pferd*, *cavalo*).
- **Duality**: A finite number of phonemes can combine to form an infinite number of words.
- **Discreteness**: The sound elements are perceived categorically, despite actually being continuous.
- **Productivity**: We can produce and understand a nearly infinite number of messages.

A little bit about languages

Some distinctive features of language:

- **Arbitrariness**: There is no inherent relationship between form and meaning (e.g. *horse*, *pferd*, *cavalo*).
- **Duality**: A finite number of phonemes can combine to form an infinite number of words.
- **Discreteness**: The sound elements are perceived categorically, despite actually being continuous.
- **Productivity**: We can produce and understand a nearly infinite number of messages.

A little bit about languages

Some distinctive features of language:

- **Arbitrariness**: There is no inherent relationship between form and meaning (e.g. *horse*, *pferd*, *cavalo*).
- **Duality**: A finite number of phonemes can combine to form an infinite number of words.
- **Discreteness**: The sound elements are perceived categorically, despite actually being continuous.
- **Productivity**: We can produce and understand a nearly infinite number of messages.

Natural Language Processing

Computational Linguistics models various aspects of languages:

1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

Natural Language Processing

Computational Linguistics models various aspects of languages:

1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

Natural Language Processing

Computational Linguistics models various aspects of languages:

1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

Natural Language Processing

Computational Linguistics models various aspects of languages:

1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

Natural Language Processing

Computational Linguistics models various aspects of languages:

1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

Natural Language Processing

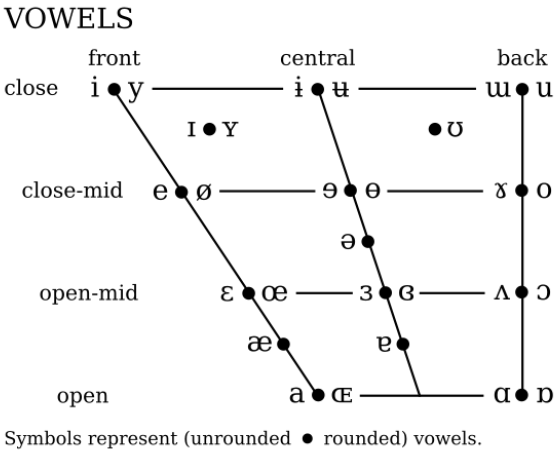
Computational Linguistics models various aspects of languages:

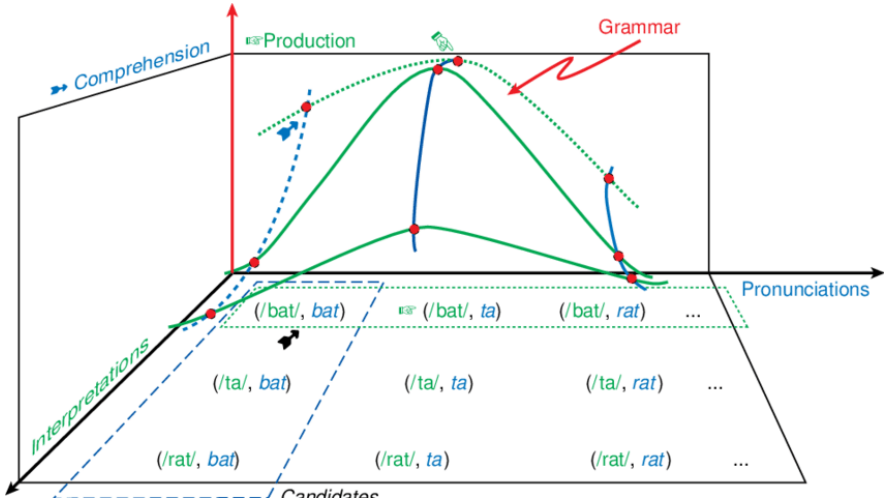
1. **Phonetics/Phonology** –> Sound formation.
2. **Morphology** –> Word formation.
3. **Syntax** –> Sentence formation.
4. **Semantics/Pragmatics** –> Meaning assignment.

NLP is more oriented towards building solutions with computers.

What will not talk about today

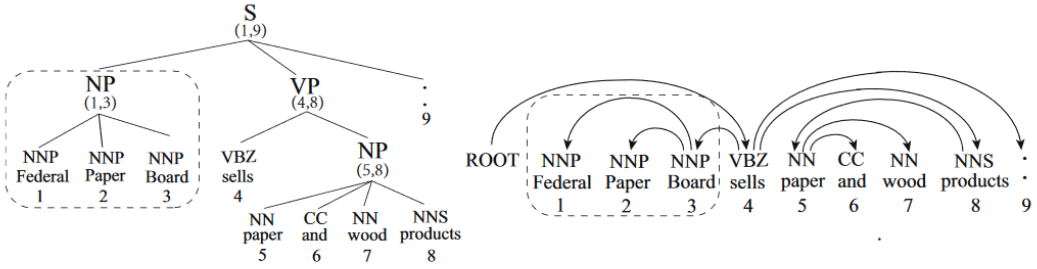
Articulatory, acoustic and auditory phonetics / phonology.



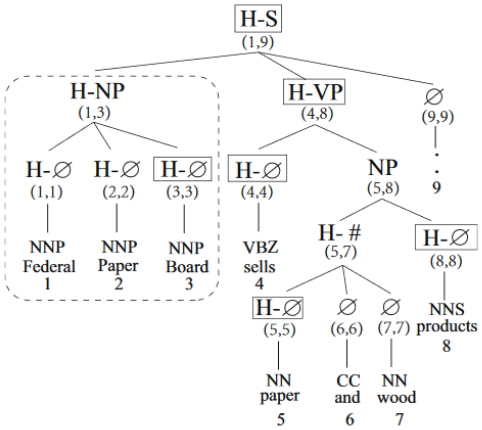


What will not talk about today

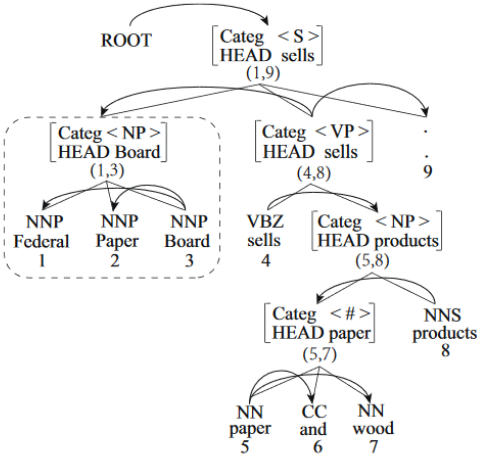
Syntactic theories and formal grammars.



(a) Constituent and dependency.



(b) Division span structure.



(c) Joint span structure.

What will not talk about today

Formal semantics, interpretability and meaning representations.

A man owns a book.

$$
\dfrac{
\dfrac{
\dfrac{\text{a}}{\begin{array}{c} S/(S\backslash NP)/N \\ : \lambda n.\lambda P.\begin{bmatrix} w : \begin{bmatrix} y : \textbf{entity} \\ ny \end{bmatrix} \\ p(\pi_1 w) \end{bmatrix} \end{array}}
\quad
\dfrac{\text{man}}{\begin{array}{c} N \\ : \textbf{man} \end{array}}
}{\begin{array}{c} S/(S\backslash NP) \\ : \lambda P.\begin{bmatrix} w : \begin{bmatrix} y : \textbf{entity} \\ \textbf{man}(y) \end{bmatrix} \\ p(\pi_1 w) \end{bmatrix} \end{array}} >
\quad
\dfrac{
\dfrac{\text{owns}}{\begin{array}{c} S\backslash NP/NP \\ : \lambda y.\lambda x.\textbf{own}(x, y) \end{array}}
\quad
\dfrac{
\dfrac{\text{a}}{\begin{array}{c} S\backslash NP\backslash(S\backslash NP/NP)/N \\ : \lambda n.\lambda P.\lambda x.\begin{bmatrix} u : \begin{bmatrix} y : \textbf{entity} \\ ny \end{bmatrix} \\ p(\pi_1 u)x \end{bmatrix} \end{array}}
\quad
\dfrac{\text{book}}{\begin{array}{c} N \\ : \textbf{book} \end{array}}
}{\begin{array}{c} S\backslash NP\backslash(S\backslash NP/NP) \\ : \lambda P.\lambda x\begin{bmatrix} u : \begin{bmatrix} y : \textbf{entity} \\ \textbf{book}(y) \end{bmatrix} \\ p(\pi_1 u)x \end{bmatrix} \end{array}} >
}{\begin{array}{c} S\backslash NP \\ : \lambda x.\begin{bmatrix} u : \begin{bmatrix} y : \textbf{entity} \\ \textbf{book}(y) \end{bmatrix} \\ \textbf{own}(x, \pi_1 u) \end{bmatrix} \end{array}} <
}{\begin{array}{c} S \\ : \begin{bmatrix} w : \begin{bmatrix} y : \textbf{entity} \\ \textbf{man}(y) \end{bmatrix} \\ \begin{bmatrix} u : \begin{bmatrix} y : \textbf{entity} \\ \textbf{book}(y) \end{bmatrix} \\ \textbf{own}(\pi_1 w, \pi_1 u) \end{bmatrix} \end{bmatrix} \end{array}} >
$$

What will not talk about today

Many other theoretical aspects of the linguistic study.

What will not talk about today

Many other theoretical aspects of the linguistic study.

... but I encourage you to read about this!

What will we talk about

Recent approaches for certain tasks that have been successful in real-world NLP applications.

- **Featurization**: How to turn strings into tractable mathematical objects.
- **Machine translation**: How to convert a string in one language to a string in another language that conveys a similar message.
- **Text classification**: How to distinguish texts from different authors or about different topics.
- **Transformers**: What are these new models everyone is talking about.

What will we talk about

Recent approaches for certain tasks that have been successful in real-world NLP applications.

- **Featurization**: How to turn strings into tractable mathematical objects.
- **Machine translation**: How to convert a string in one language to a string in another language that conveys a similar message.
- **Text classification**: How to distinguish texts from different authors or about different topics.
- **Transformers**: What are these new models everyone is talking about.

What will we talk about

Recent approaches for certain tasks that have been successful in real-world NLP applications.

- **Featurization**: How to turn strings into tractable mathematical objects.
- **Machine translation**: How to convert a string in one language to a string in another language that conveys a similar message.
- **Text classification**: How to distinguish texts from different authors or about different topics.
- **Transformers**: What are these new models everyone is talking about.

What will we talk about

Recent approaches for certain tasks that have been successful in real-world NLP applications.

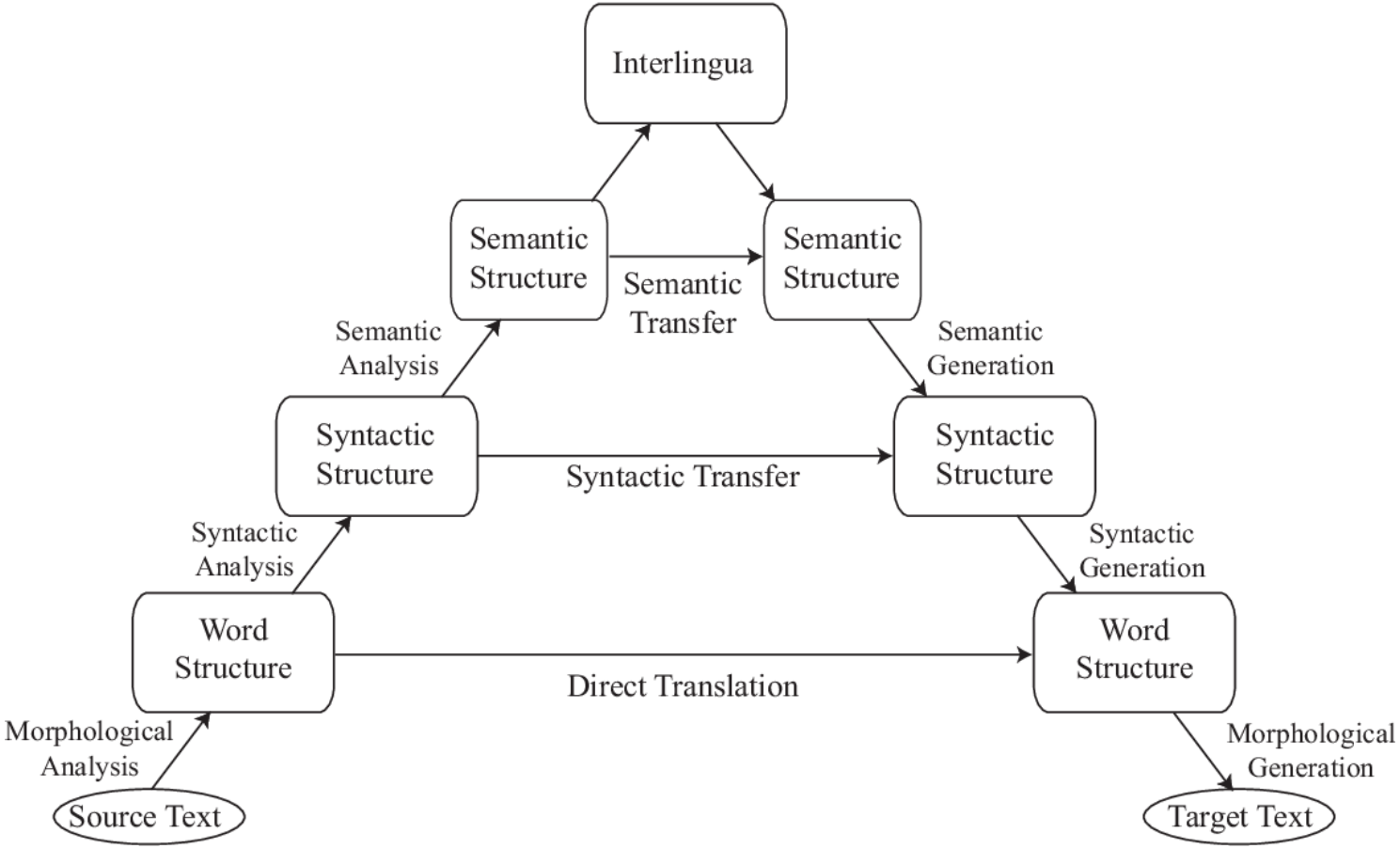
- **Featurization**: How to turn strings into tractable mathematical objects.
- **Machine translation**: How to convert a string in one language to a string in another language that conveys a similar message.
- **Text classification**: How to distinguish texts from different authors or about different topics.
- **Transformers**: What are these new models everyone is talking about.

What will we talk about

Recent approaches for certain tasks that have been successful in real-world NLP applications.

- **Featurization**: How to turn strings into tractable mathematical objects.
- **Machine translation**: How to convert a string in one language to a string in another language that conveys a similar message.
- **Text classification**: How to distinguish texts from different authors or about different topics.
- **Transformers**: What are these new models everyone is talking about.

What have we accomplished?



What have we accomplished?

The majority of commercially successful approaches today disregard syntax and semantics.

What have we accomplished?

The majority of commercially successful approaches today disregard syntax and semantics.

Instead they are based on some form of word-level statistical analysis or machine learning...

What have we accomplished?

The majority of commercially successful approaches today disregard syntax and semantics.

Instead they are based on some form of word-level statistical analysis or machine learning...

...but it would seem that these alone cannot really *understand* Language to a decent degree.

What have we accomplished?

The majority of commercially successful approaches today disregard syntax and semantics.

Instead they are based on some form of word-level statistical analysis or machine learning...

...but it would seem that these alone cannot really *understand* Language to a decent degree.

There is still a long road ahead, and that is good.

1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

Vector Space Model

Key idea: Everything is a **vector** in some high-dimensional space.

Vector Space Model

Key idea: Everything is a **vector** in some high-dimensional space.

What are the dimensions of that space (basis vectors)?

Vector Space Model

Key idea: Everything is a **vector** in some high-dimensional space.

What are the dimensions of that space (basis vectors)?

How to project a fragment of text to that space?

Vector Space Model

Key idea: Everything is a **vector** in some high-dimensional space.

What are the dimensions of that space (basis vectors)?

How to project a fragment of text to that space?

How to compare texts?

Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

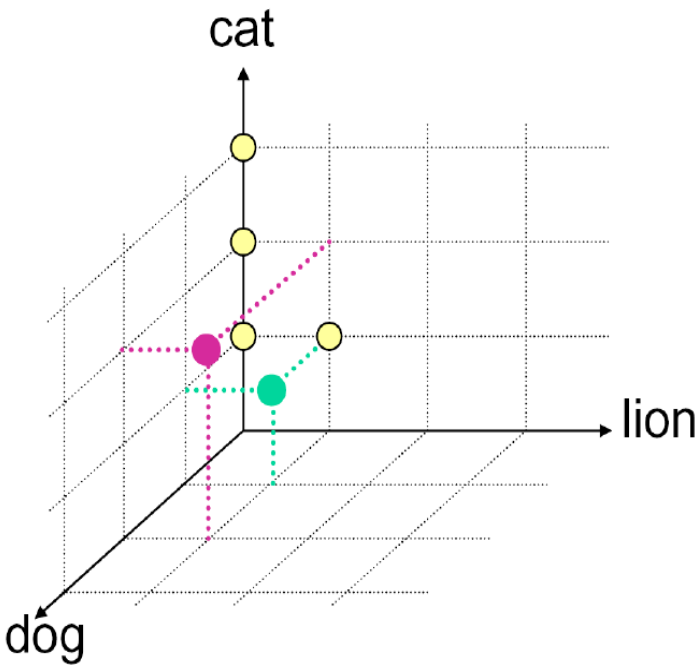
Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

Vector Space Model 1: dimensions = words

- Separate dimension for each word.
  - **words = coordinate vectors**.
  - value along dimension *cat* = times the word *cat* occurs.
- The difference between documents is the distance between points in space.
  - Euclidean distance or angle between vectors.
  - Usually expressed as similarity.

Vector Space Model 1: dimensions = words



cat = {0,0,1}

cat cat = {0,0,2}

cat cat cat = {0,0,3}

cat lion = {1,0,1}

cat lion dog = {1,1,1}

cat cat lion dog dog = {1,2,2}

Dot product

Similarity of document vector $\mathbf{A}$ and $\mathbf{B}$ is the distance between points in space.

Dot product

Similarity of document vector $\mathbf{A}$ and $\mathbf{B}$ is the distance between points in space.

$$\mathbf{A} = [a_1, a_2, ..., a_d] \quad \mathbf{B} = [b_1, b_2, ..., b_d]$$

Dot product

Similarity of document vector $\mathbf{A}$ and $\mathbf{B}$ is the distance between points in space.

$$\mathbf{A} = [a_1, a_2, ..., a_d] \quad \mathbf{B} = [b_1, b_2, ..., b_d]$$

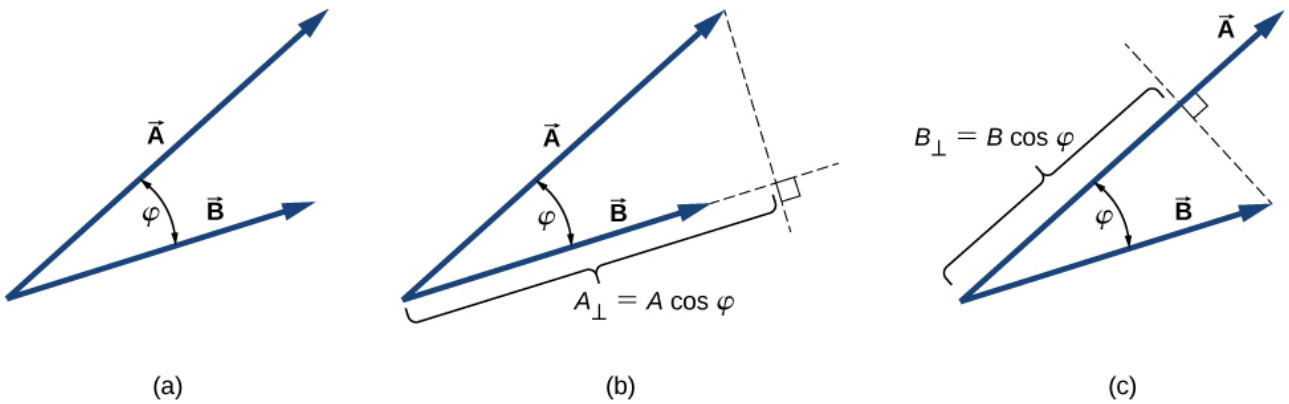
$$\mathbf{A}^\top \mathbf{B} = \sum_{i=0}^{d} a_i b_i = \|A\| \, \|B\| \cos(\varphi)$$

Dot product

Similarity of document vector $\mathbf{A}$ and $\mathbf{B}$ is the distance between points in space.

$$\mathbf{A} = [a_1, a_2, \ldots, a_d] \quad \mathbf{B} = [b_1, b_2, \ldots, b_d]$$

$$\mathbf{A}^\top \mathbf{B} = \sum_{i=0}^{d} a_i b_i = \|A\| \, \|B\| \cos(\varphi)$$



Dot product

Geometric interpretation: length of the projection of $\mathbf{A}$ onto $\mathbf{B}$.

- highest if the two vectors point in the same direction.
- zero if the two vectors are orthogonal (no words in common).

Dot product

Geometric interpretation: length of the projection of $\mathbf{A}$ onto $\mathbf{B}$.

- highest if the two vectors point in the same direction.
- zero if the two vectors are orthogonal (no words in common).

Dot product

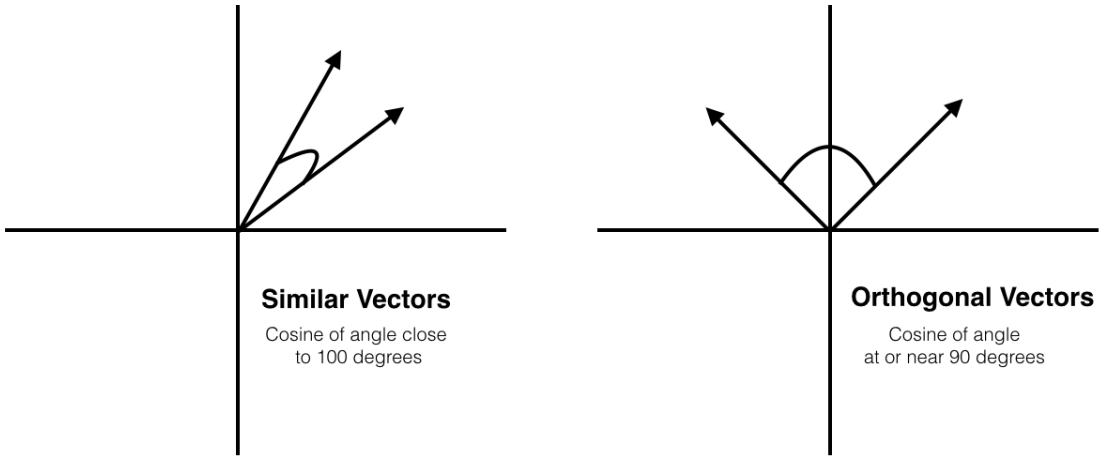
Geometric interpretation: length of the projection of $\mathbf{A}$ onto $\mathbf{B}$.

- highest if the two vectors point in the same direction.
- zero if the two vectors are orthogonal (no words in common).

Dot product

Geometric interpretation: length of the projection of $\mathbf{A}$ onto $\mathbf{B}$.

- highest if the two vectors point in the same direction.
- zero if the two vectors are orthogonal (no words in common).

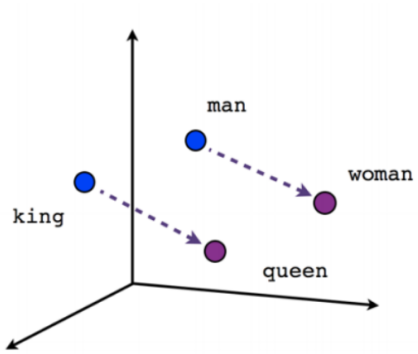


Vector Space Model 2: word2vec

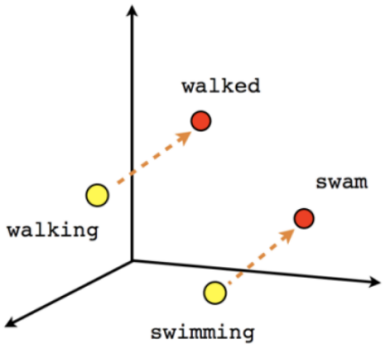
The traditional model has a very limited notion of similarity, and the vectors are extremely sparse.

Vector Space Model 2: word2vec

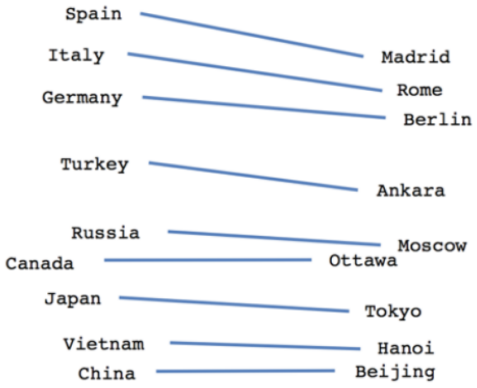
The traditional model has a very limited notion of similarity, and the vectors are extremely sparse.



Male-Female



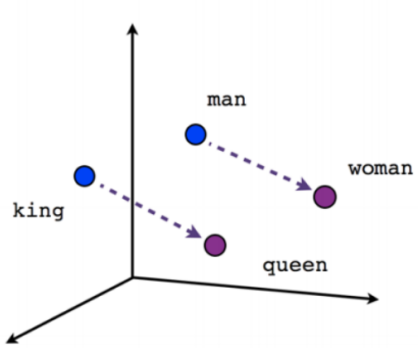
Verb tense



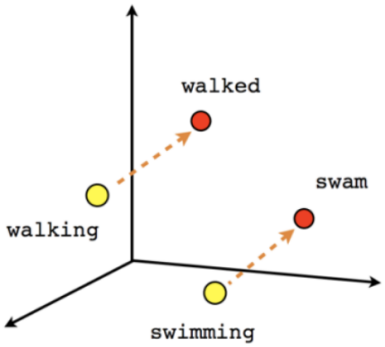
Country-Capital

Vector Space Model 2: word2vec

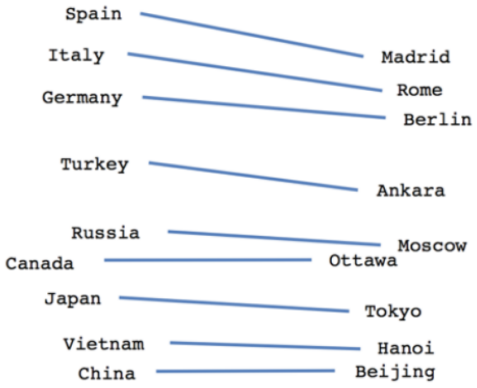
The traditional model has a very limited notion of similarity, and the vectors are extremely sparse.



Male-Female



Verb tense



Country-Capital

We can come up with a better representation called **word2vec**.

Vector Space Model 2: word2vec

Principle of distributional semantics

The meaning of a word is determined by the words it occurs together with.

Vector Space Model 2: word2vec

Principle of distributional semantics

The meaning of a word is determined by the words it occurs together with.

If that is true, it follows that we can distribute the meaning of a word across dimensions.

Skip-gram model

The **skip-gram model** can help us produce the vector representations we need.

Skip-gram model

The **skip-gram model** can help us produce the vector representations we need.

We can learn it by using a neural network.

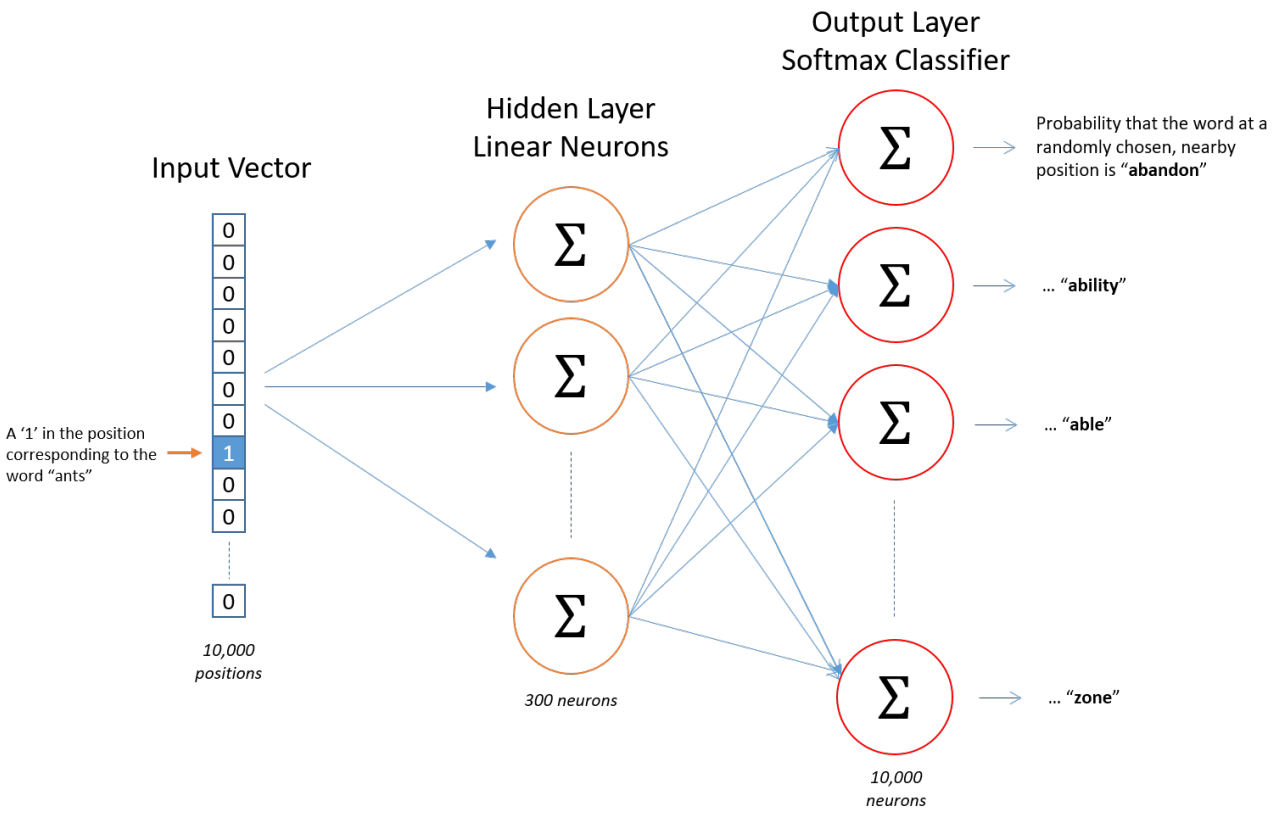
Skip-gram model

The **skip-gram model** can help us produce the vector representations we need.

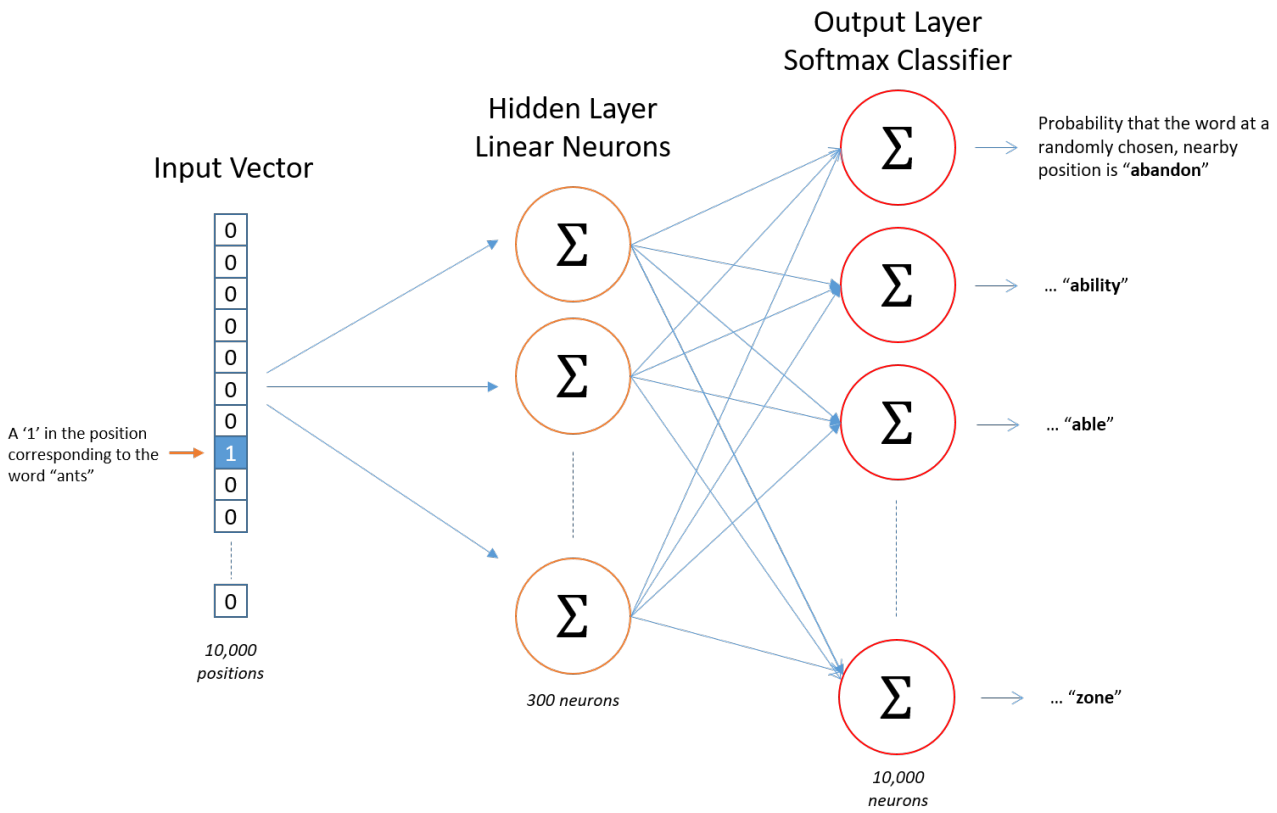
We can learn it by using a neural network.

Task: Given a specific word in the middle of a sentence, infer which word that is likely to appear nearby.

Skip-gram network



Skip-gram network



The weights in the hidden layer are our 300D vectors.

Skip-gram network: intuition

If two words have similar contexts, the network would need to output similar results.

Skip-gram network: intuition

If two words have similar contexts, the network would need to output similar results.

The only way for the network to do this, is if the word vectors for the two words are made similar.

Skip-gram network: training

We can easily collect a training set.

- Input: one-hot vector.
- Output: one-hot vector.

Skip-gram network: training

We can easily collect a training set.

- Input: one-hot vector.
- Output: one-hot vector.

Skip-gram network: training

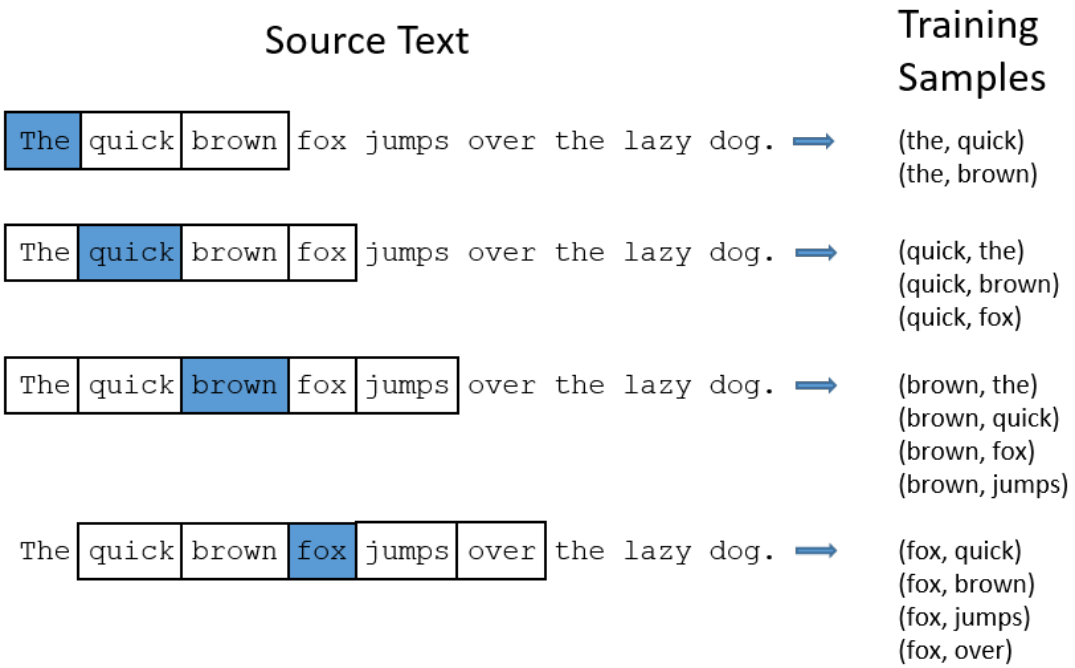
We can easily collect a training set.

- Input: one-hot vector.
- Output: one-hot vector.

Skip-gram network: training

We can easily collect a training set.

- Input: one-hot vector.
- Output: one-hot vector.

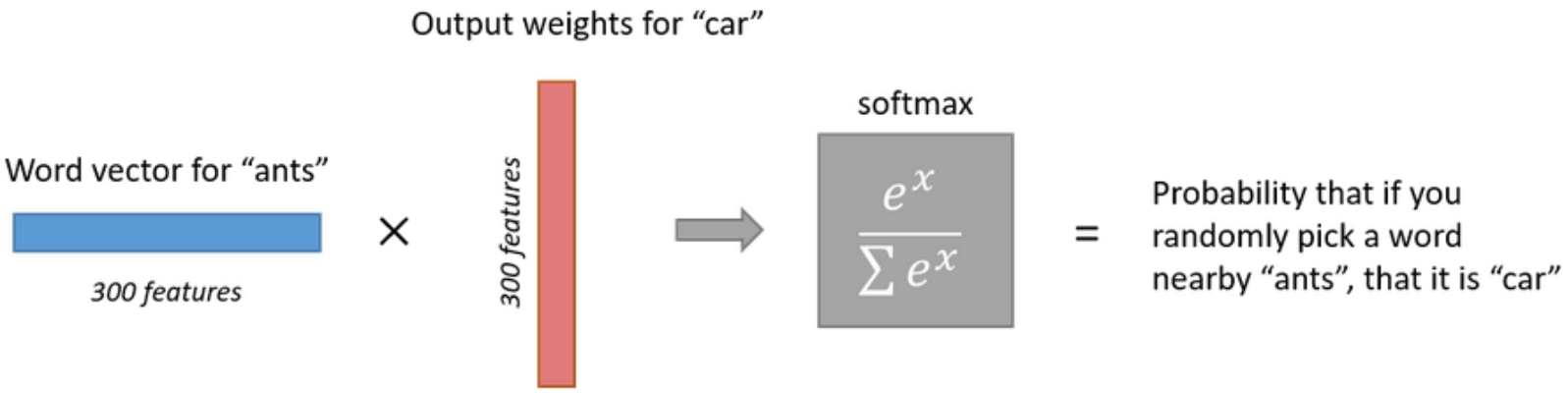


Skip-gram network: training

The output layer of the skip-gram network squashes the vector representations into a probability distribution via **softmax**.

Skip-gram network: training

The output layer of the skip-gram network squashes the vector representations into a probability distribution via **softmax**.



Skip-gram network: training

The resulting skip-gram network has many parameters, so we needs some tricks to learn it quickly.

- Subsample frequent words to decrease the number of training examples.
- Each training sample can only update a small percentage of the network weights, something called **negative sampling**.

Skip-gram network: training

The resulting skip-gram network has many parameters, so we needs some tricks to learn it quickly.

- Subsample frequent words to decrease the number of training examples.
- Each training sample can only update a small percentage of the network weights, something called **negative sampling**.

Skip-gram network: training

The resulting skip-gram network has many parameters, so we needs some tricks to learn it quickly.

- Subsample frequent words to decrease the number of training examples.
- Each training sample can only update a small percentage of the network weights, something called **negative sampling**.

Problems with word2vec

The vector for the word *bank* is the same for all sentences.

- He cashed a check in the *bank*.
- They pulled the canoe up on the *bank*.

Problems with word2vec

The vector for the word *bank* is the same for all sentences.

- He cashed a check in the *bank*.
- They pulled the canoe up on the *bank*.

Problems with word2vec

The vector for the word *bank* is the same for all sentences.

- He cashed a check in the *bank*.
- They pulled the canoe up on the *bank*.

Problems with word2vec

The vector for the word *bank* is the same for all sentences.

- He cashed a check in the *bank*.
- They pulled the canoe up on the *bank*.

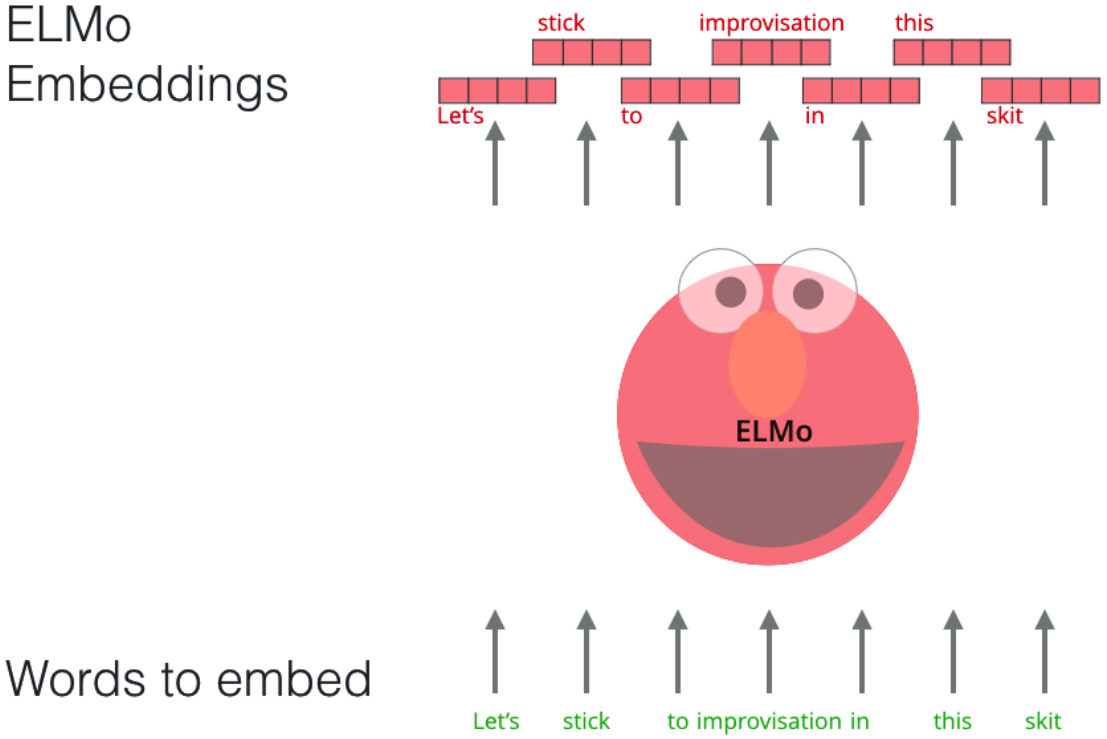
Context matters for meaning. One single word form can have many different meanings.

Vector Space Model 3: ELMo

Instead of using a fixed embedding for each word, **ELMo** looks at the entire sentence before assigning an embedding.

Vector Space Model 3: ELMo

Instead of using a fixed embedding for each word, **ELMo** looks at the entire sentence before assigning an embedding.



Why can ELMo exist?

ELMo learns word representations through building bidirectional language models.

Language modeling

The tasks which attempts to predict the next word in a sentence, or in other words, determine how likely a given sentence is.

Why can ELMo exist?

ELMo learns word representations through building bidirectional language models.

Language modeling

The tasks which attempts to predict the next word in a sentence, or in other words, determine how likely a given sentence is.

Why can ELMo exist?

ELMo learns word representations through building bidirectional language models.

Language modeling

The tasks which attempts to predict the next word in a sentence, or in other words, determine how likely a given sentence is.

Forward:

$$p(w_1, ..., w_N) = \prod_{k=1}^{N} p(w_k | w_1, ..., w_{k-1})$$

Why can ELMo exist?

ELMo learns word representations through building bidirectional language models.

Language modeling

The tasks which attempts to predict the next word in a sentence, or in other words, determine how likely a given sentence is.

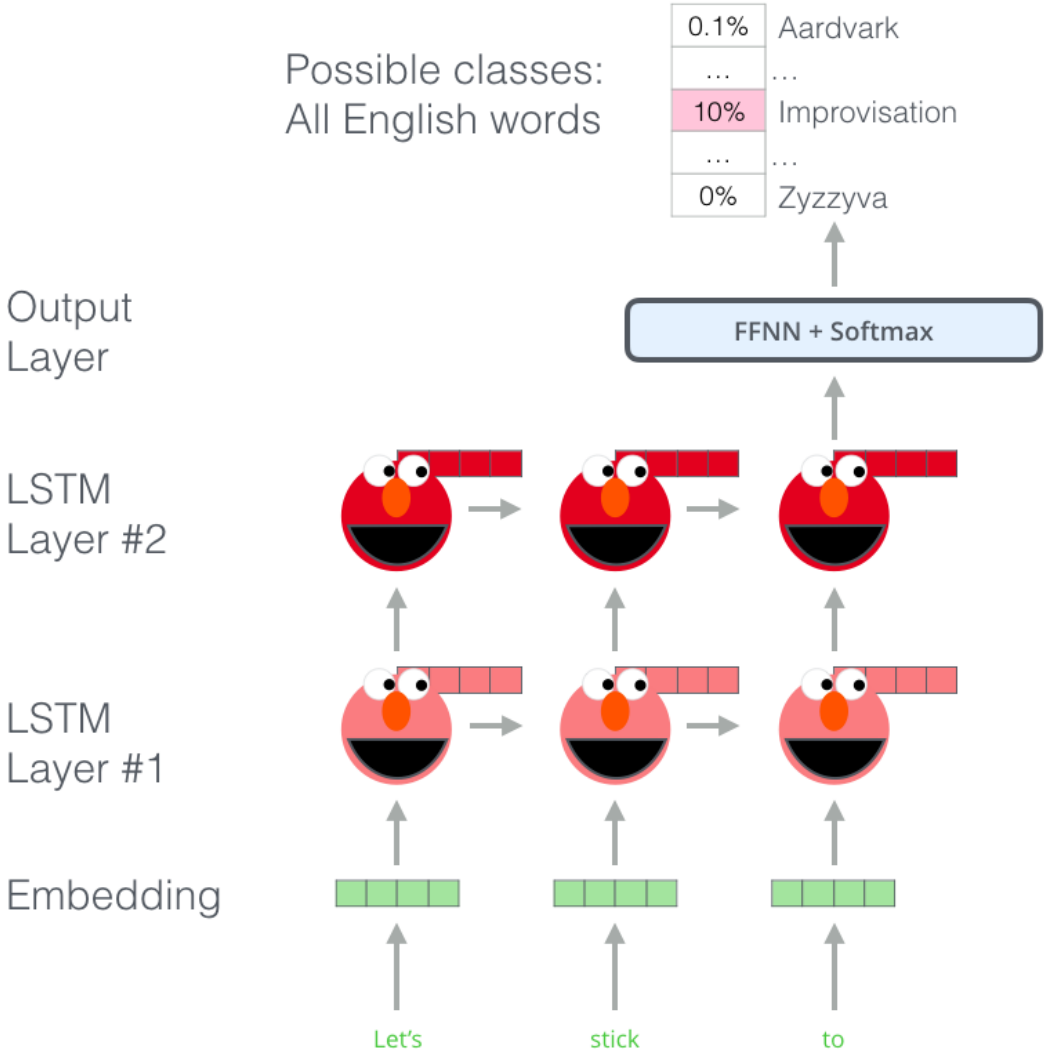
Forward:

$$p(w_1, \ldots, w_N) = \prod_{k=1}^{N} p(w_k | w_1, \ldots, w_{k-1})$$

Backward:

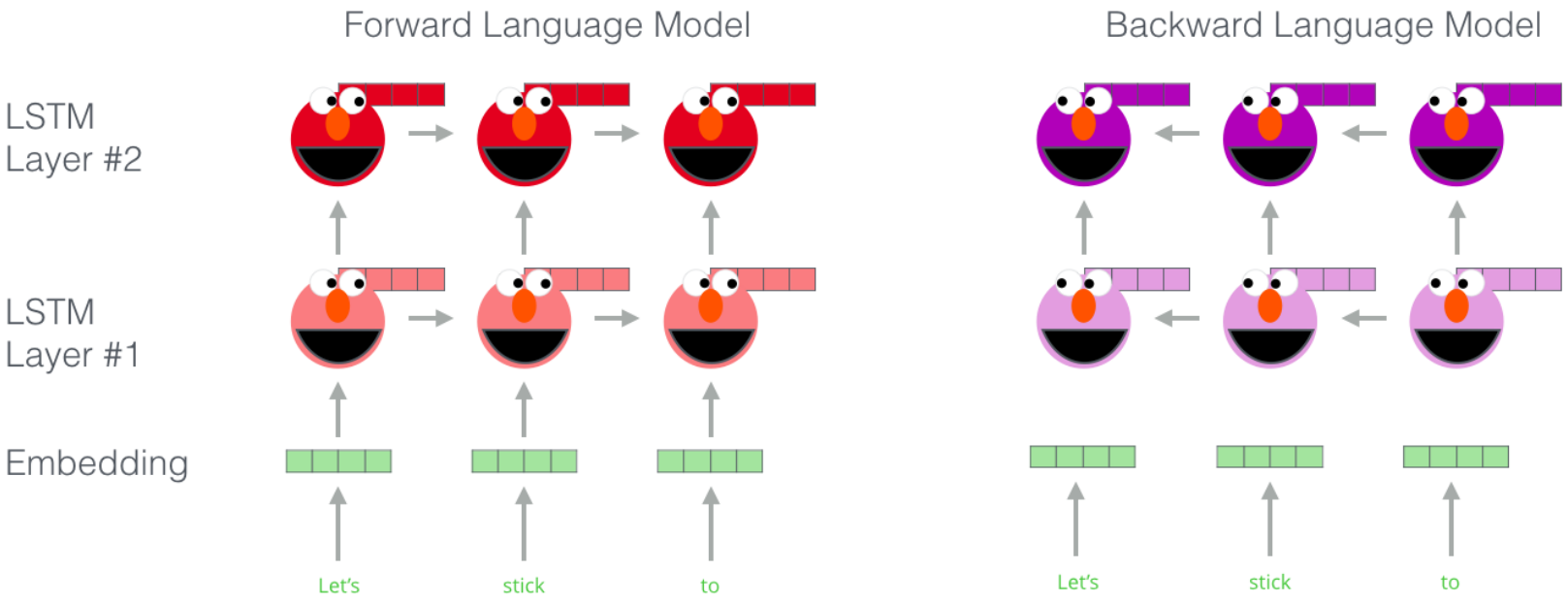
$$p(w_1, \ldots, w_N) = \prod_{k=1}^{N} p(w_k | w_{k+1}, \ldots, w_N)$$

Unidirectional Language Model

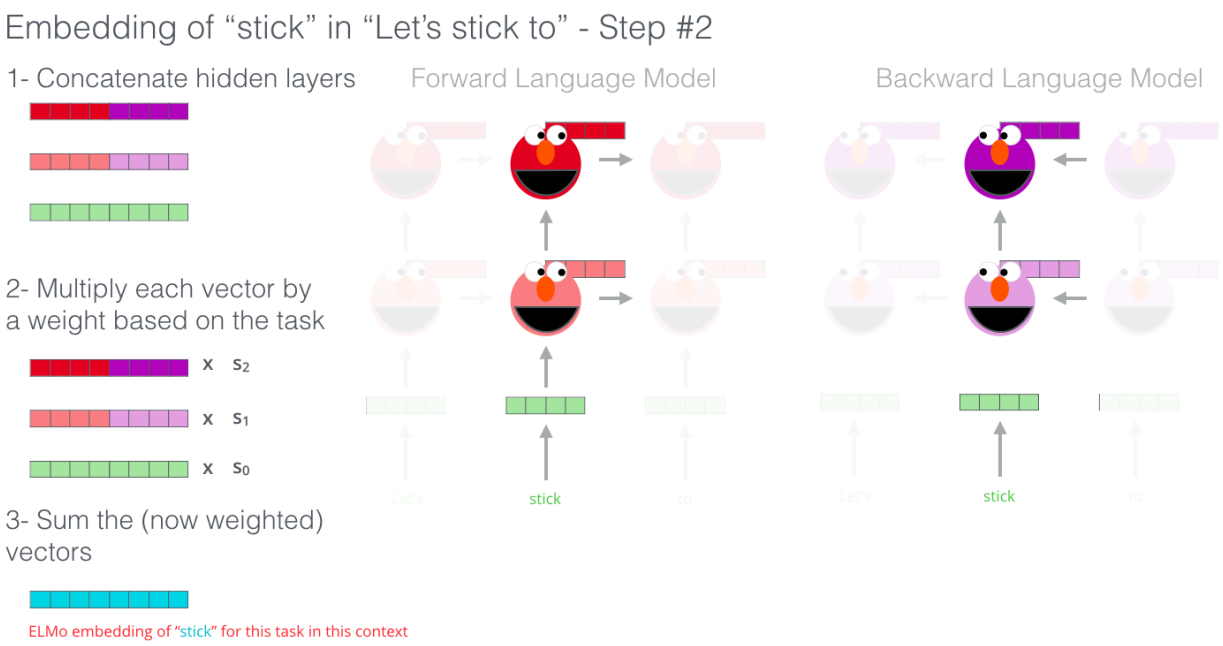


Bidirectional Language Model

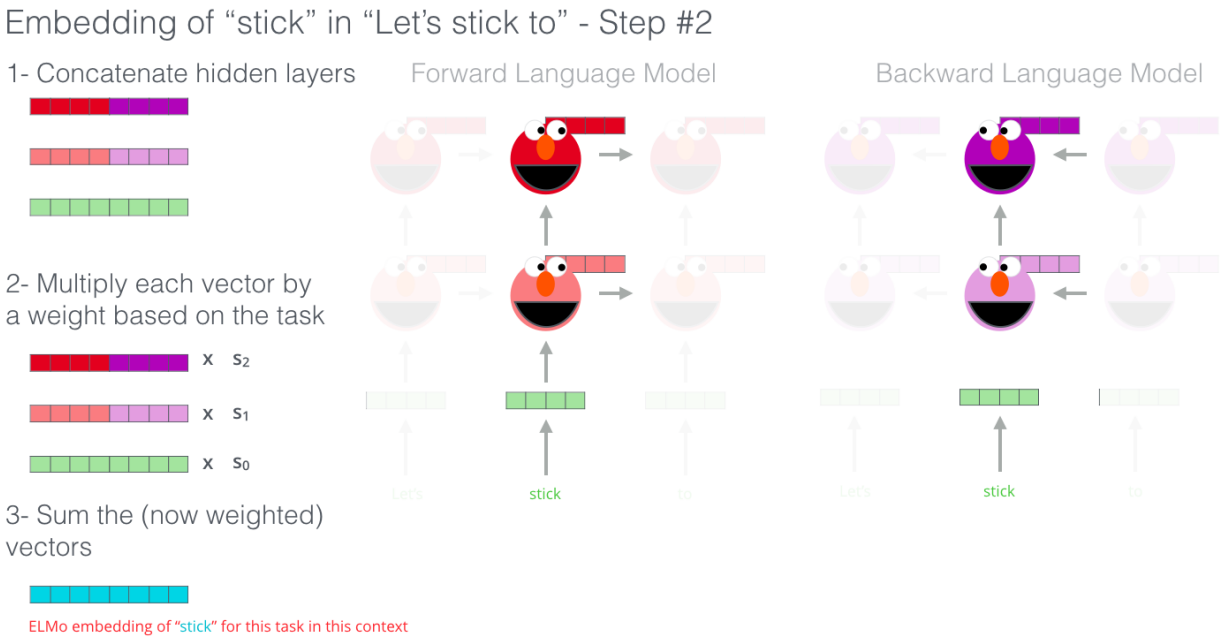
Embedding of “stick” in “Let’s stick to” - Step #1



ELMo representations

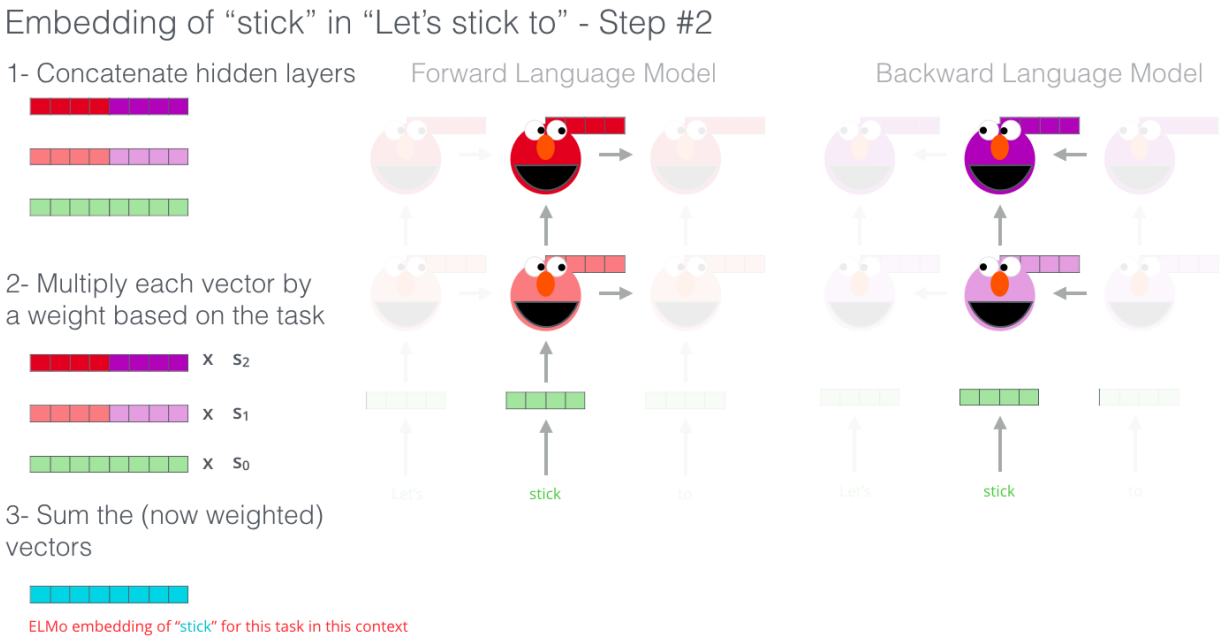


ELMo representations



$$\text{ELMO}_k^{task} = \gamma_k(s_0^{task} x_k + s_1^{task} h_{1,k} + s_2^{task} h_{2,k})$$

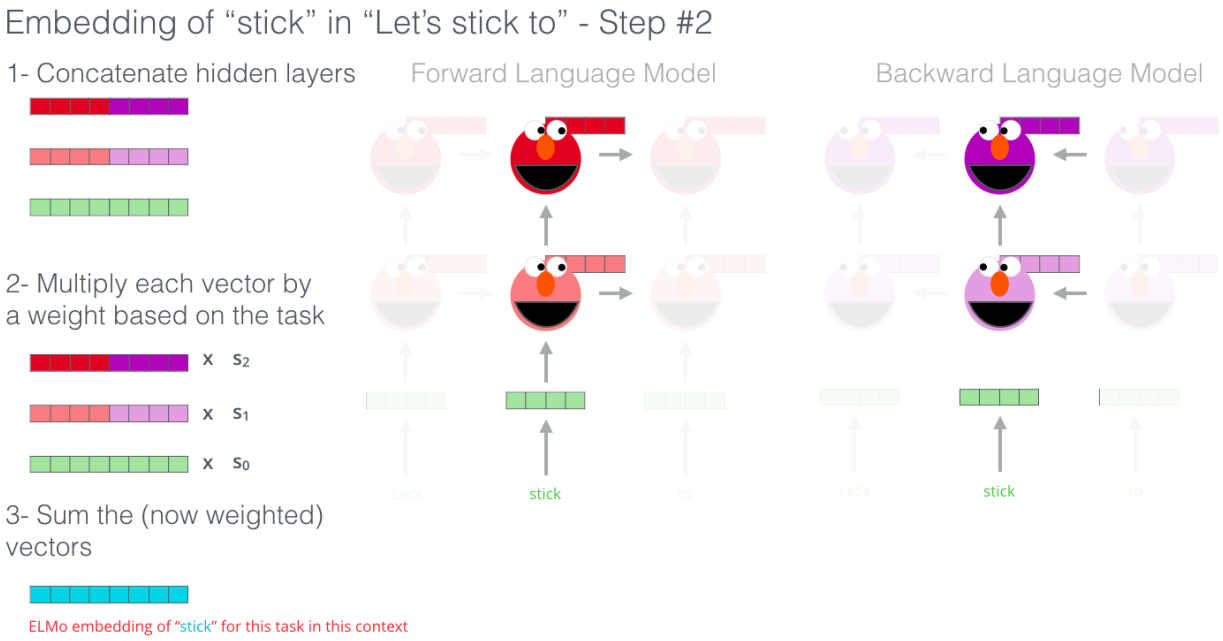
ELMo representations



$$\text{ELMO}_k^{task} = \gamma_k(s_0^{task} x_k + s_1^{task} h_{1,k} + s_2^{task} h_{2,k})$$

s_i^{task} : softmax-normalized weights from the hidden layers.

ELMo representations



$$\text{ELMO}_k^{task} = \gamma_k(s_0^{task} x_k + s_1^{task} h_{1,k} + s_2^{task} h_{2,k})$$

s_i^{task} : softmax-normalized weights from the hidden layers.

γ_k : task specific scaling factor.

ELMo representations

The factors γ_k and s_i are then learned while training for a certain task (e.g. question answering)

ELMo representations

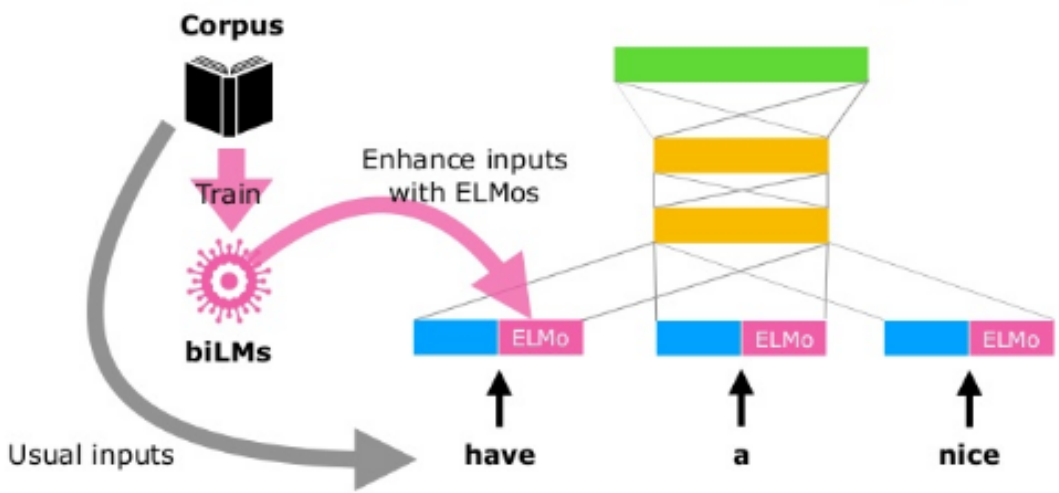
The factors γ_k and s_i are then learned while training for a certain task (e.g. question answering)

Different ELMo vectors for each task!

ELMo representations

The factors γ_k and s_i are then learned while training for a certain task (e.g. question answering)

Different ELMo vectors for each task!



1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

Machine Translation systems

Machine translation can be treated as cryptography.

Machine Translation systems

Machine translation can be treated as cryptography.

There are three predominant types:

Machine Translation systems

Machine translation can be treated as cryptography.

There are three predominant types:

- Rule-based translation.
- Statistical translation.
- Neural translation.

Machine Translation systems

Machine translation can be treated as cryptography.

There are three predominant types:

- Rule-based translation.
- Statistical translation.
- Neural translation.

Machine Translation systems

Machine translation can be treated as cryptography.

There are three predominant types:

- Rule-based translation.
- Statistical translation.
- Neural translation.

Machine Translation systems

Machine translation can be treated as cryptography.

There are three predominant types:

- Rule-based translation.
- Statistical translation.
- Neural translation.

Learning translation models requires parallel corpora.

Lexical alignment

Given e (English word) and f (French word), we want to learn a model $P(e|f)$.

.

Lexical alignment

Given e (English word) and f (French word), we want to learn a model $P(e|f)$.

This is also a naive translation for sentences, or $P(\mathbf{e}|\mathbf{f})$.

.

Lexical alignment

Given e (English word) and f (French word), we want to learn a model $P(e|f)$.

This is also a naive translation for sentences, or $P(\mathbf{e}|\mathbf{f})$.

Linked to the idea of **word alignments**.

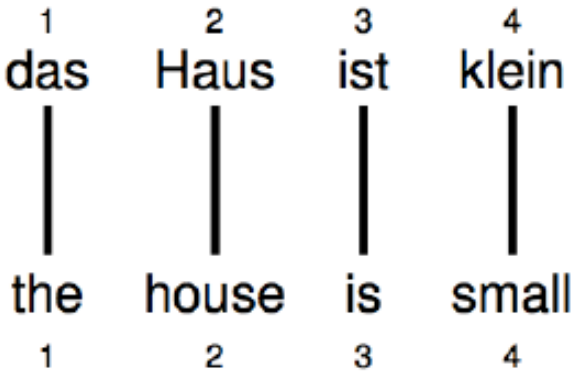
.

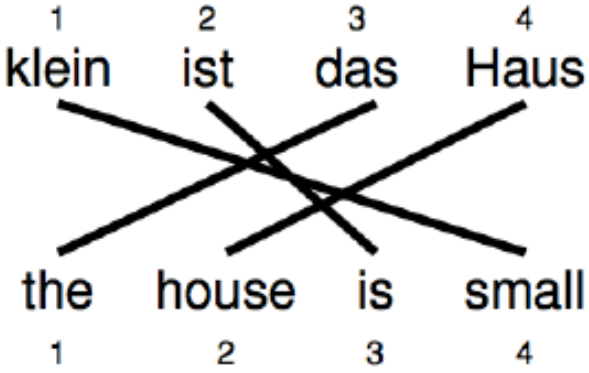
Lexical alignment

Alignments can be visualized by drawing links, and are represented as vector positions.

Lexical alignment

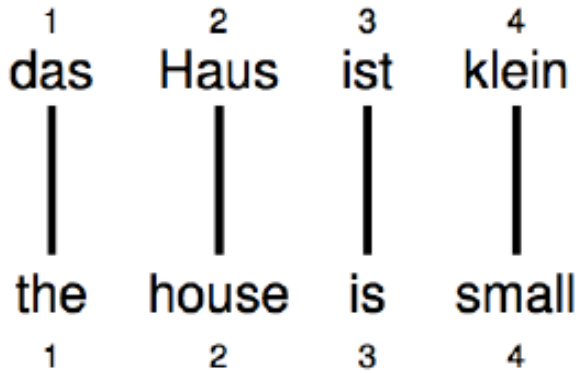
Alignments can be visualized by drawing links, and are represented as vector positions.

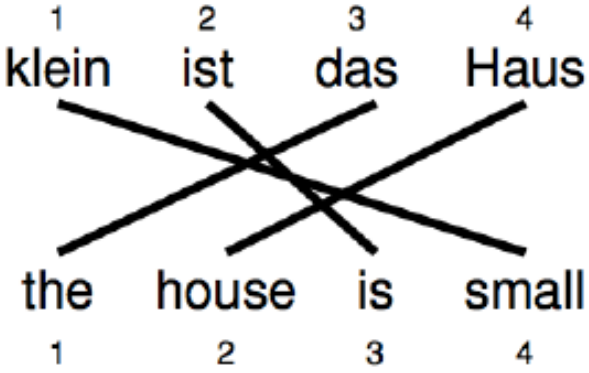


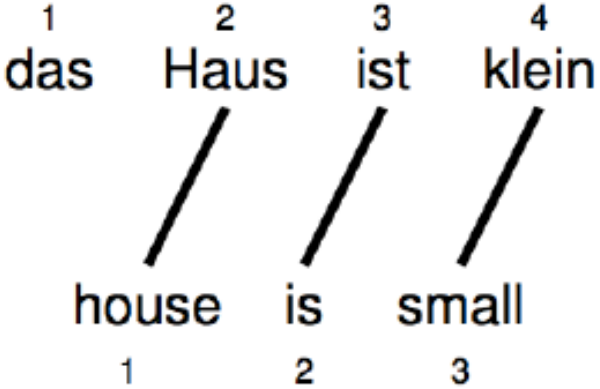


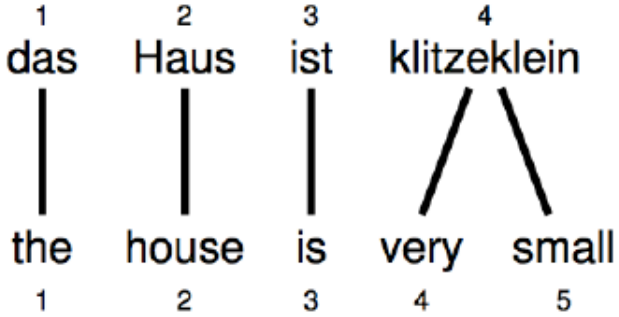
Lexical alignment

Alignments can be visualized by drawing links, and are represented as vector positions.







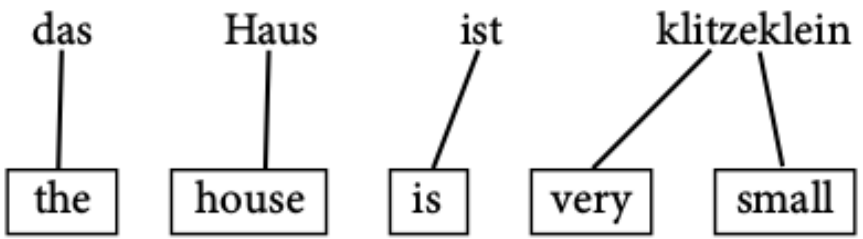


Statistical models

Generative story: Given Foreign string $\mathbf{f}$ and length m of English string, alignments $\mathbf{a}$ and English string $\mathbf{e}$ are generated.

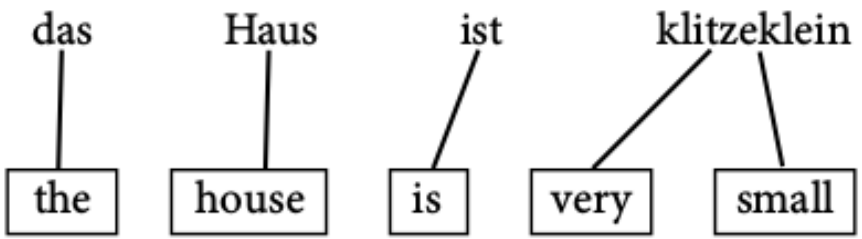
Statistical models

Generative story: Given Foreign string $\mathbf{f}$ and length m of English string, alignments $\mathbf{a}$ and English string $\mathbf{e}$ are generated.



Statistical models

Generative story: Given Foreign string $\mathbf{f}$ and length m of English string, alignments $\mathbf{a}$ and English string $\mathbf{e}$ are generated.



Model $P(\mathbf{a}, \mathbf{e}|\mathbf{f}, \mathbf{m}) = \mathbf{P}(\mathbf{e}|\mathbf{a}, \mathbf{f}, \mathbf{m}) * \mathbf{P}(\mathbf{a}|\mathbf{f}, \mathbf{m})$.

IBM Model 1

- The alignment decisions for the m English words are pairwise independent.
- The alignment distribution for each $\mathbf{a_i}$ is uniform over all source words and NULL.
- The English words are generated independently, conditioned only on their aligned Foreign words.

IBM Model 1

- The alignment decisions for the m English words are pairwise independent.
- The alignment distribution for each $\mathbf{a_i}$ is uniform over all source words and NULL.
- The English words are generated independently, conditioned only on their aligned Foreign words.

IBM Model 1

- The alignment decisions for the m English words are pairwise independent.
- The alignment distribution for each $\mathbf{a_i}$ is uniform over all source words and NULL.
- The English words are generated independently, conditioned only on their aligned Foreign words.

IBM Model 1

$$\text{for each } i \in [1, 2, \ldots, m]$$
$$a_i \sim \text{Uniform}(0, 1, 2, \ldots, n)$$
$$e_i \sim \text{Categorical}(\boldsymbol{\theta}_{f_{a_i}})$$

$$P(e_i, a_i \mid \mathbf{f}, m) = P(a_i \mid \mathbf{f}, m) \cdot P(e_i \mid a_i, \mathbf{f}, m) = \frac{1}{n+1} \cdot P(e_i \mid f_{a_i})$$

$$P(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m) = \prod_{i=1}^{m} P(e_i, a_i \mid \mathbf{f}, m) = \prod_{i=1}^{m} \frac{1}{n+1} \cdot P(e_i \mid f_{a_i})$$

$$P(\mathbf{e} \mid \mathbf{f}, m) = \sum_{\mathbf{a}} P(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m)$$

IBM Model 1 example

das

e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus

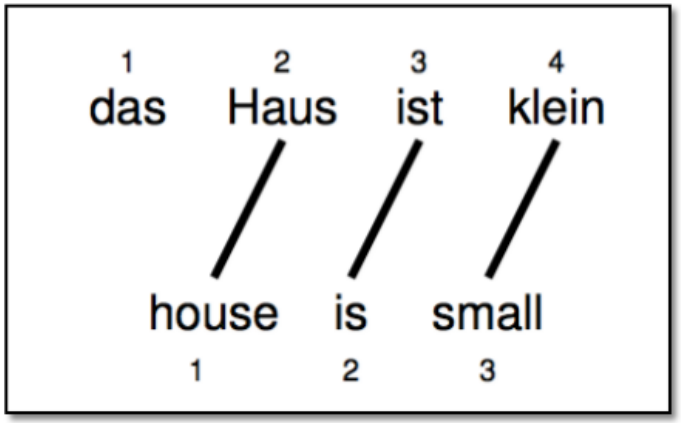
e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist

e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein

e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04



$\mathbf{a} = (2, 3, 4)$

$P(\mathbf{e}, \mathbf{a} \mid \mathbf{f}, m) =$
$1/5 * P(\text{Haus}|\text{house}) *$
$1/5 * P(\text{ist}|\text{is}) *$
$1/5 * P(\text{small}|\text{klein})$
$= 1/125 * 0.8 * 0.8 * 0.4$
$= 0.002$

Computing the best alignment

$$\mathbf{a}^* = \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} p(\mathbf{a} \mid \mathbf{e}, \mathbf{f})$$

$$= \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} \frac{p(\mathbf{e}, \mathbf{a} \mid \mathbf{f})}{\sum_{\mathbf{a}'} p(\mathbf{e}, \mathbf{a}' \mid \mathbf{f})}$$

$$= \arg \max_{\mathbf{a} \in [0,1,\ldots,n]^m} p(\mathbf{e}, \mathbf{a} \mid \mathbf{f})$$

a_i all independent

$$a_i^* = \arg \max_{a_i=0}^{n} \frac{1}{1+n} p(e_i \mid f_{a_i})$$

$$= \arg \max_{a_i=0}^{n} p(e_i \mid f_{a_i})$$

IBM Model 1 extensions

- IBM Model 2: $P(\mathbf{a})$ not uniform, but implements reordering model that prefers alignments in which words stay close to their original position.
- IBM Model 3: adds fertility model that predicts the number of English words to which a given $\mathbf{f}$ will be aligned.
- IBM Models 4-5: more complicated reordering models.

IBM Model 1 extensions

- IBM Model 2: $P(\mathbf{a})$ not uniform, but implements reordering model that prefers alignments in which words stay close to their original position.
- IBM Model 3: adds fertility model that predicts the number of English words to which a given $\mathbf{f}$ will be aligned.
- IBM Models 4-5: more complicated reordering models.

IBM Model 1 extensions

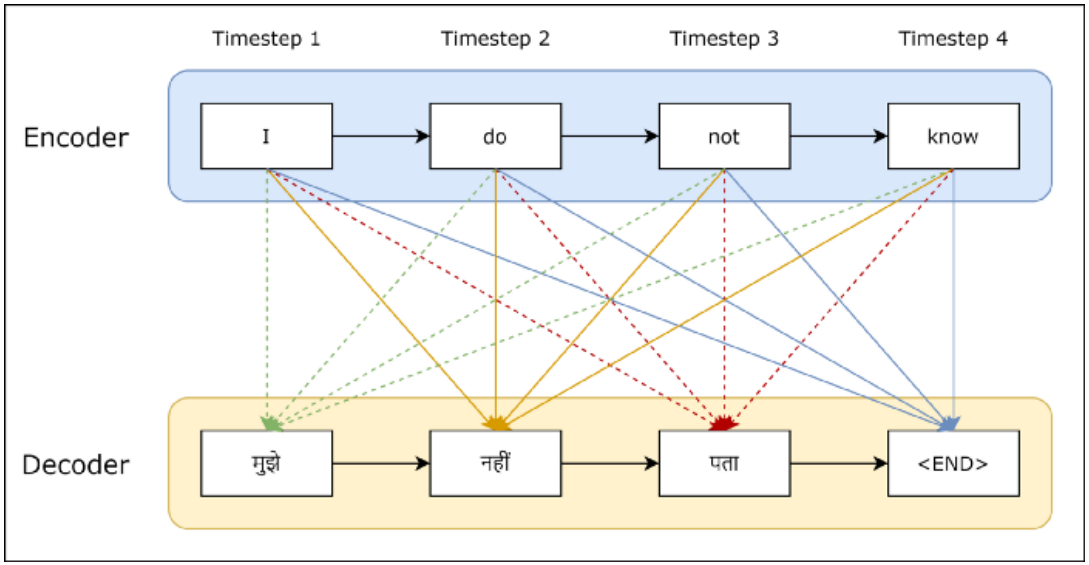
- IBM Model 2: $P(\mathbf{a})$ not uniform, but implements reordering model that prefers alignments in which words stay close to their original position.
- IBM Model 3: adds fertility model that predicts the number of English words to which a given $\mathbf{f}$ will be aligned.
- IBM Models 4-5: more complicated reordering models.

Encoder-decoder architecture

We humans read an entire sentence, understand it, and then produce a translation. This is a **seq2seq** approach with 3 parts:

Encoder-decoder architecture

We humans read an entire sentence, understand it, and then produce a translation. This is a **seq2seq** approach with 3 parts:



Neural Machine Translation (NMT)

Neural networks can very naturally be used to build encoder-decoder models.

Neural Machine Translation (NMT)

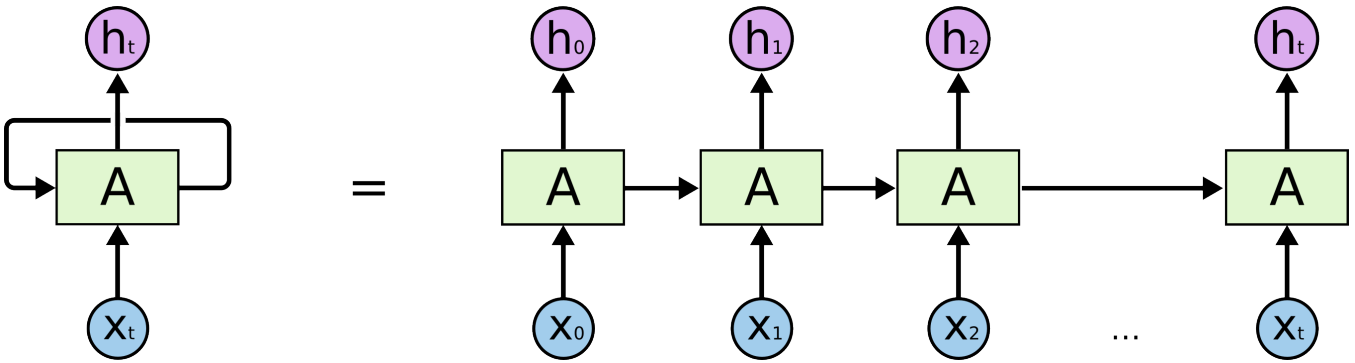
Neural networks can very naturally be used to build encoder-decoder models.

Usually, some form of **Recurrent Neural Networks** (RNN) is employed to capture sentence-wide dependencies.

Neural Machine Translation (NMT)

Neural networks can very naturally be used to build encoder-decoder models.

Usually, some form of **Recurrent Neural Networks** (RNN) is employed to capture sentence-wide dependencies.



Long distance dependencies

"*I grew up in France. [...] I speak fluent French*"

Long distance dependencies

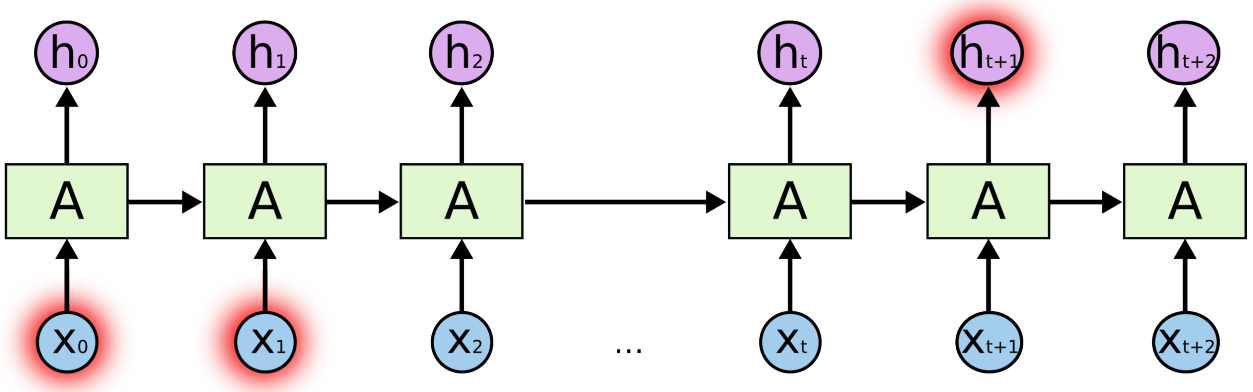
"*I grew up in France. [...] I speak fluent French*"

As the gap grows bigger, RNNs are less able to learn this information. This is called the **vanishing gradient** problem.

Long distance dependencies

"*I grew up in France. [...] I speak fluent French*"

As the gap grows bigger, RNNs are less able to learn this information. This is called the **vanishing gradient** problem.



LSTM

Long Short Term Memory networks or LSTMs are capable of attenuating the vanishing gradient problem.

LSTM

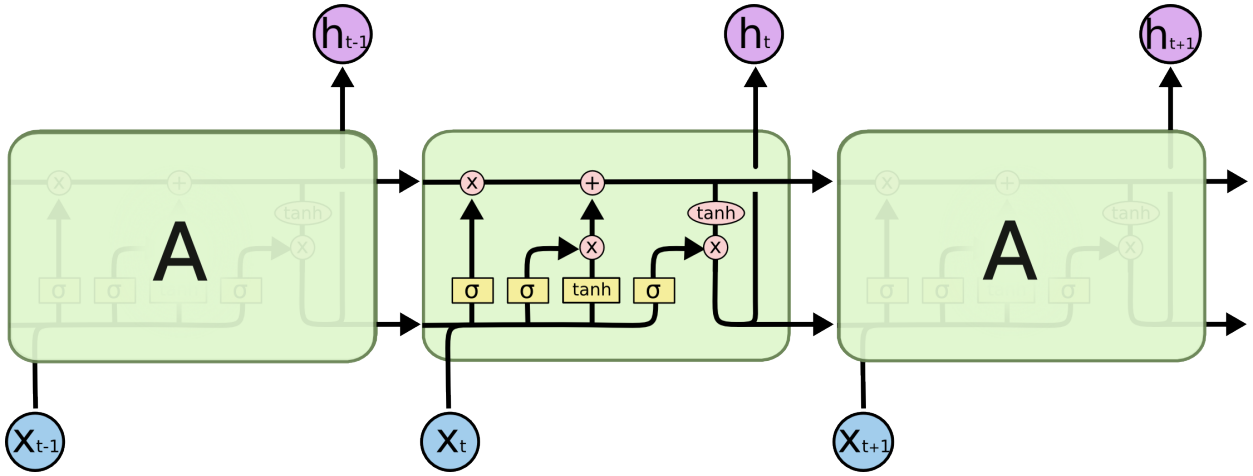
Long Short Term Memory networks or LSTMs are capable of attenuating the vanishing gradient problem.

Now each point in time in our unfolded RNN has a **cell state**.

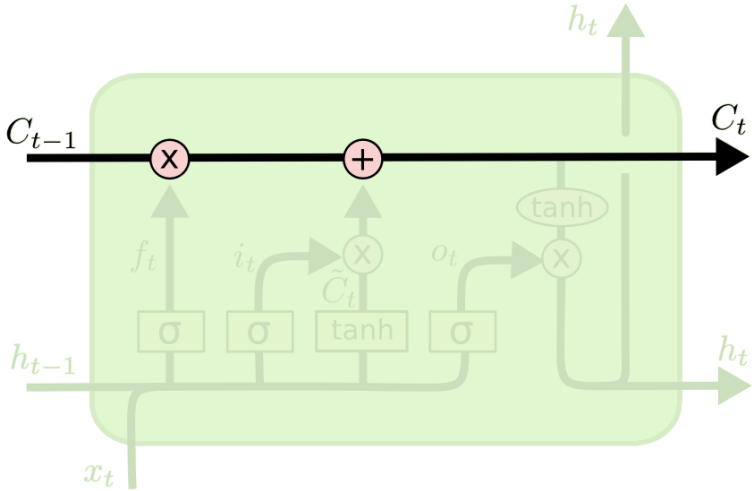
LSTM

Long Short Term Memory networks or LSTMs are capable of attenuating the vanishing gradient problem.

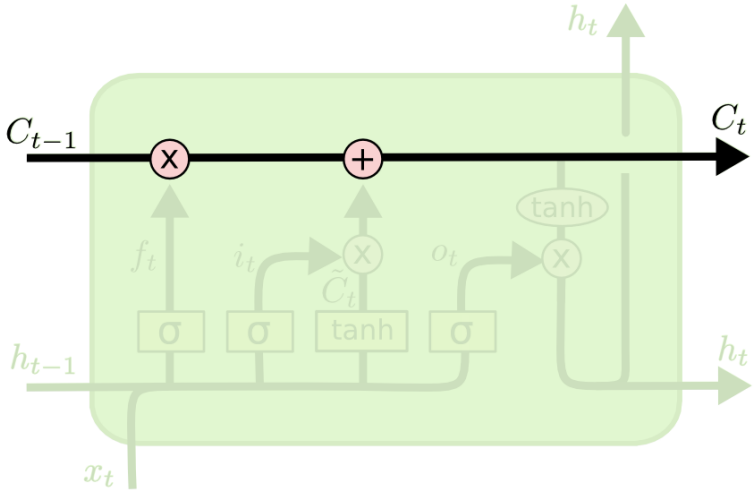
Now each point in time in our unfolded RNN has a **cell state**.



LSTM - Cell state

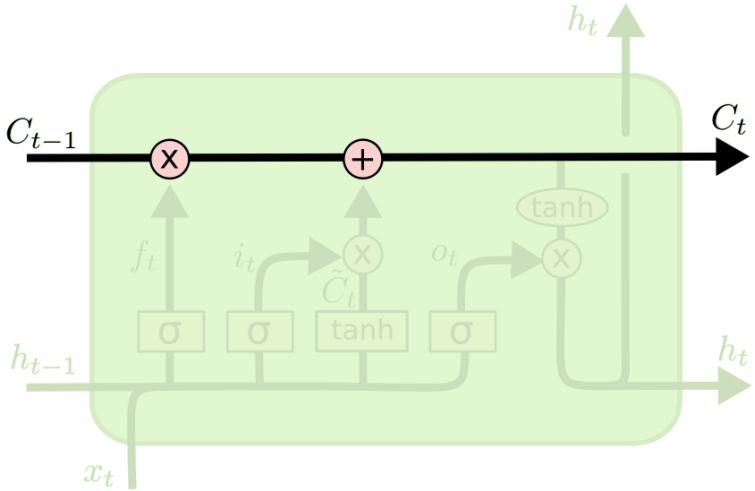


LSTM - Cell state



The **cell state** is like a conveyor belt, delivering information to the next step in the chain.

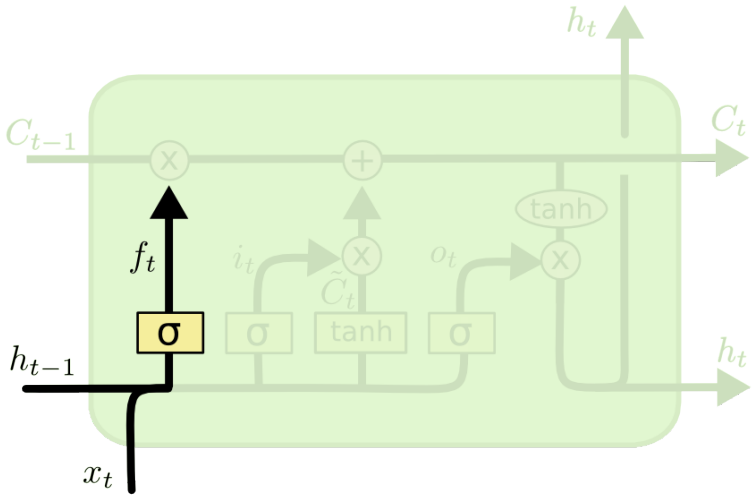
LSTM - Cell state



The **cell state** is like a conveyor belt, delivering information to the next step in the chain.

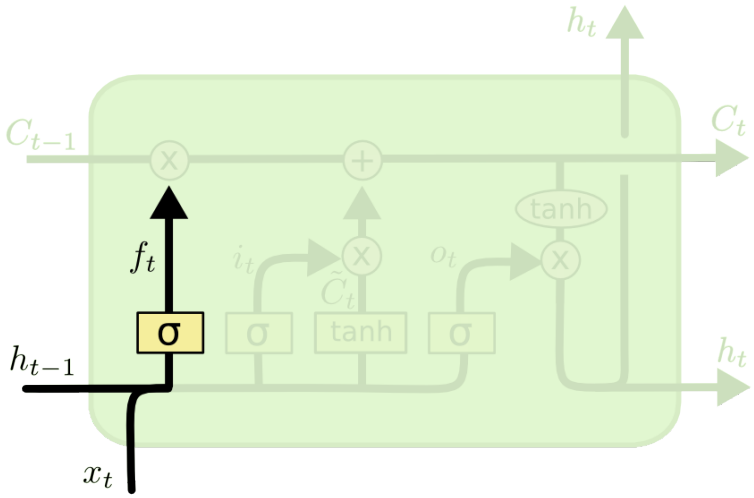
Information can be removed or added via **gated interactions**.

LSTM - Forget gate layer



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

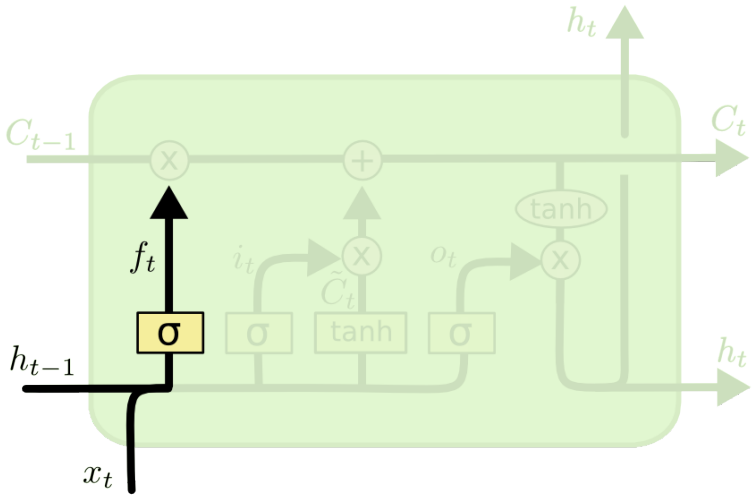
LSTM - Forget gate layer



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Decide which elements of the previous cell state to keep ($\simeq$1) or forget ($\simeq$0).

LSTM - Forget gate layer

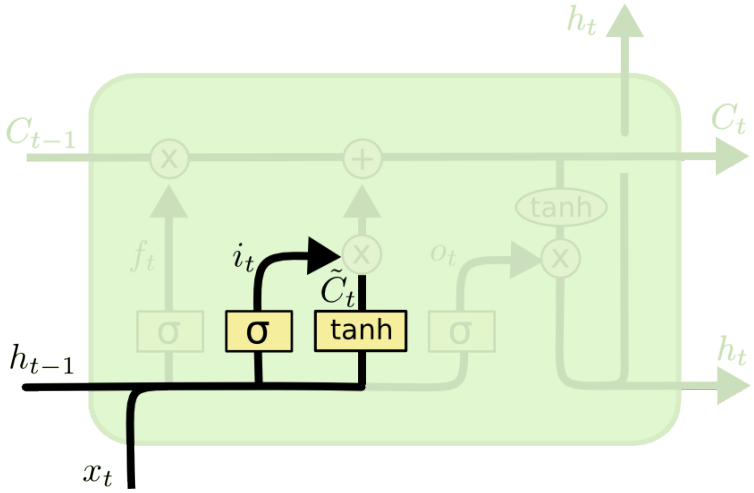


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Decide which elements of the previous cell state to keep ($\simeq$1) or forget ($\simeq$0).

Think about remembering the gender of the subject for predicting future words in language modelling.

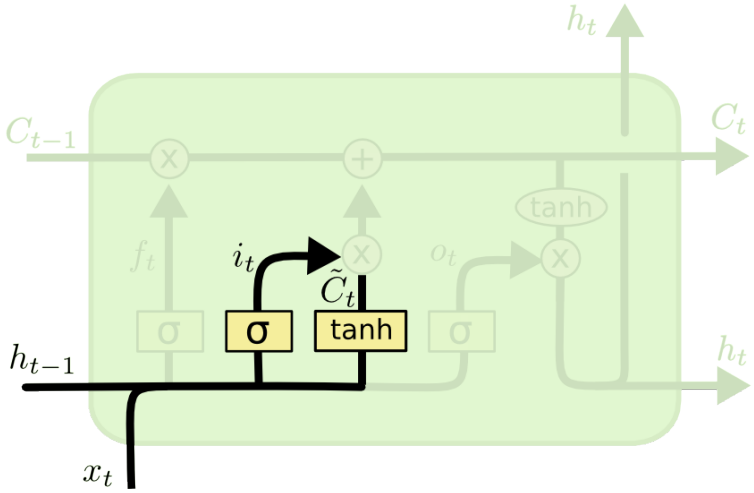
LSTM - Update gate layer



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM - Update gate layer

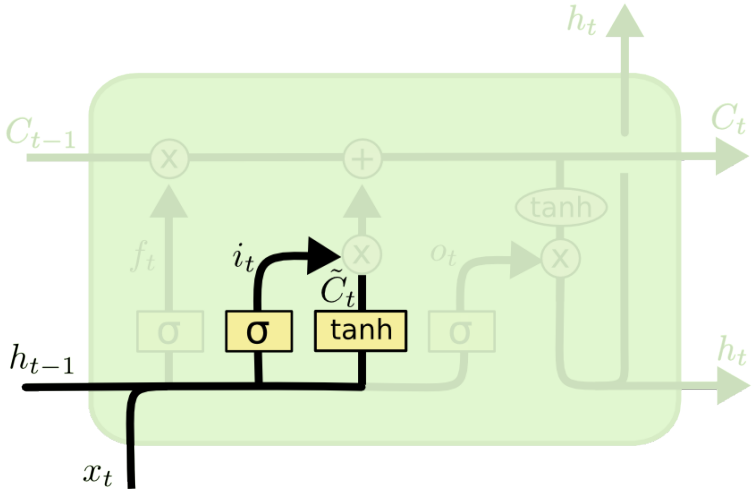


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Decide which information is to be stored in the current cell state.

LSTM - Update gate layer



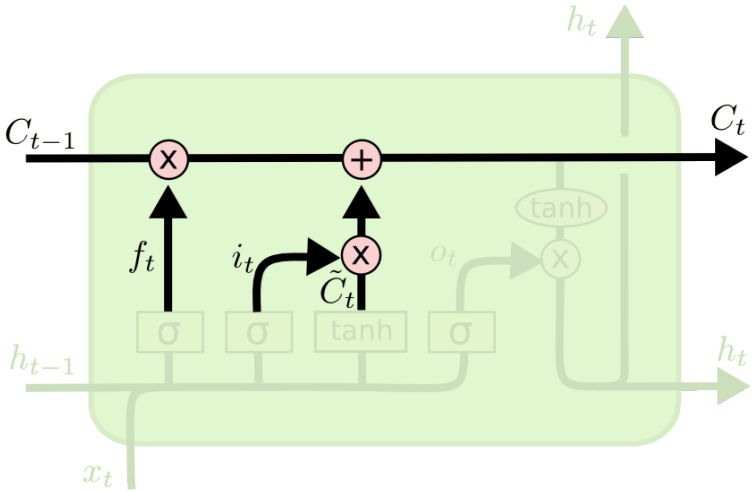
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Decide which information is to be stored in the current cell state.

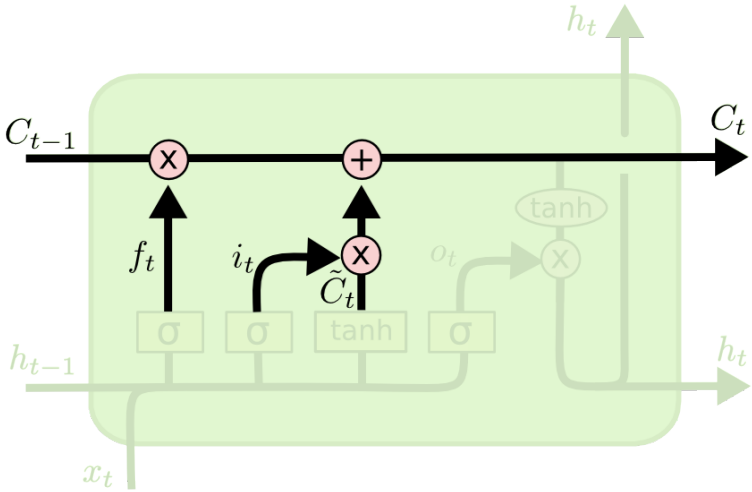
Determine which elements are relevant + produce values to add to (or subtract from) the current state.

LSTM - Update gate layer



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

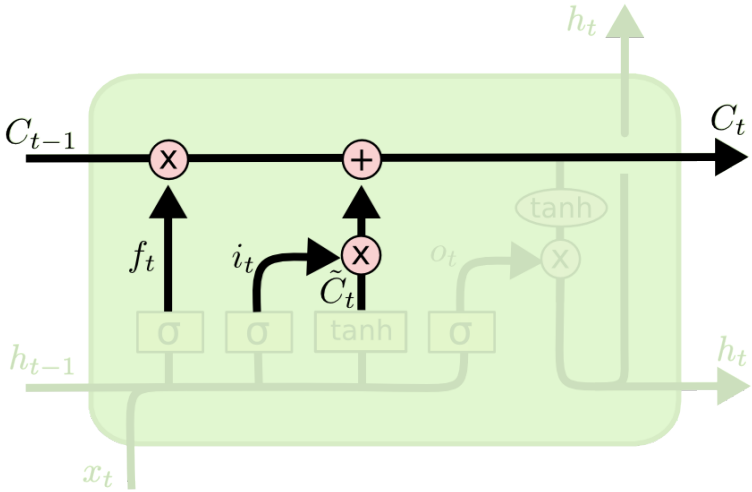
LSTM - Update gate layer



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Do the corresponding math to update the current state.

LSTM - Update gate layer

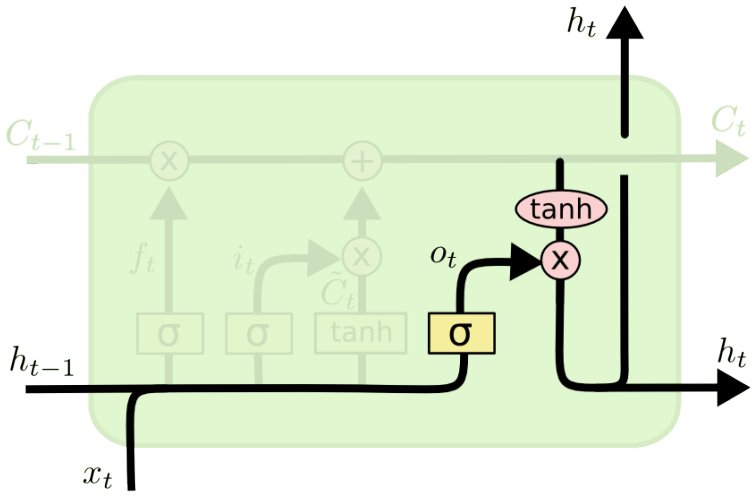


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Do the corresponding math to update the current state.

The updated state becomes the input of the next time step.

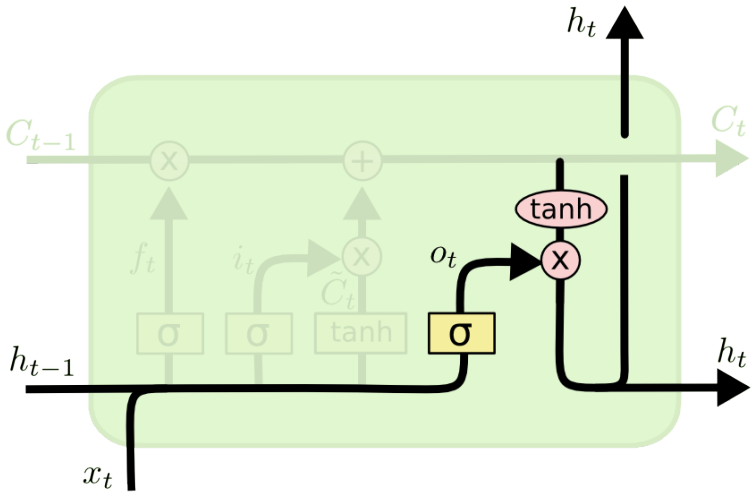
LSTM - Output gate layer



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM - Output gate layer

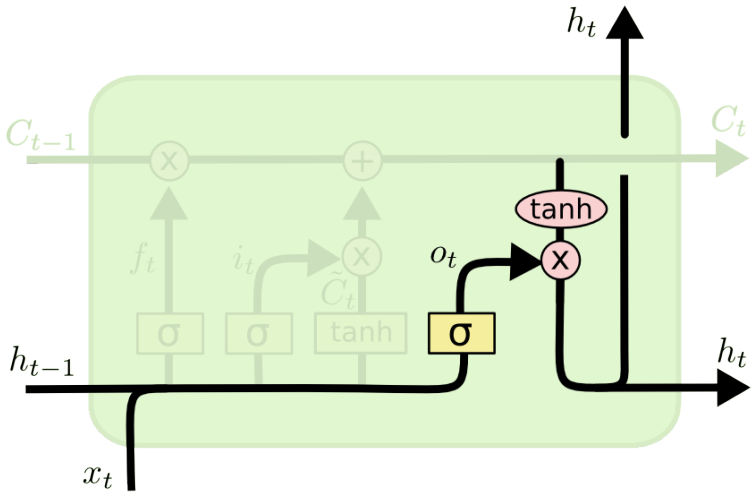


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what are we going to output at the current time step.

LSTM - Output gate layer



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what are we going to output at the current time step.

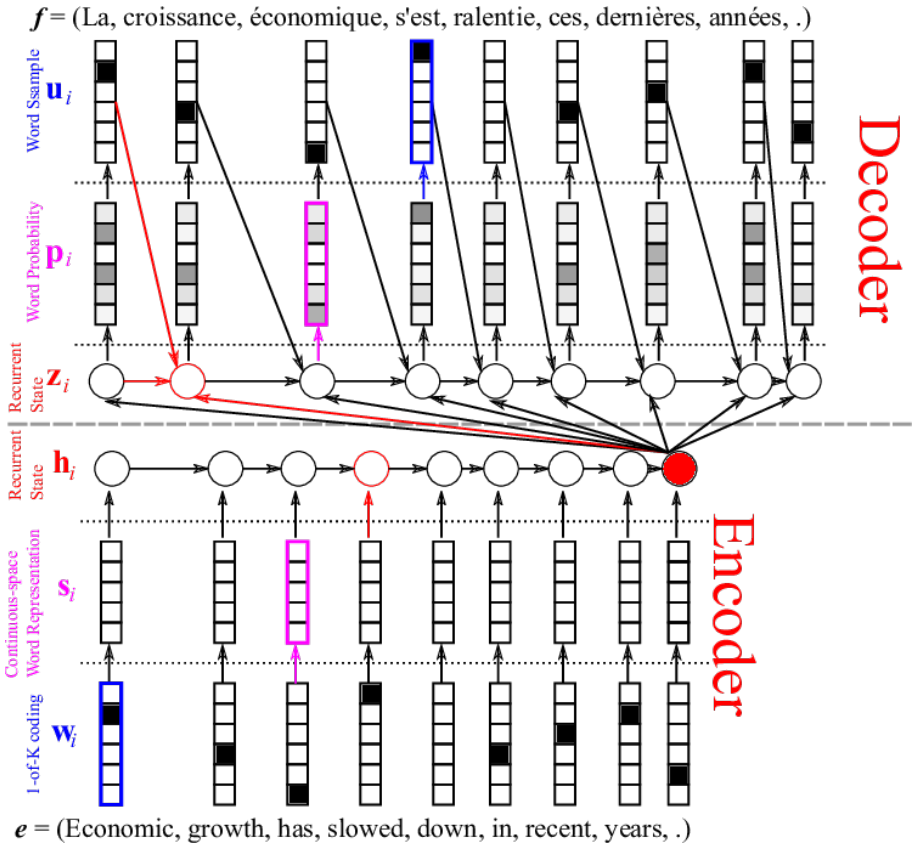
To do so, we can combine parts of the current cell state and the previous output.

Unidirectional RNNs

We can now build a working system.

Unidirectional RNNs

We can now build a working system.

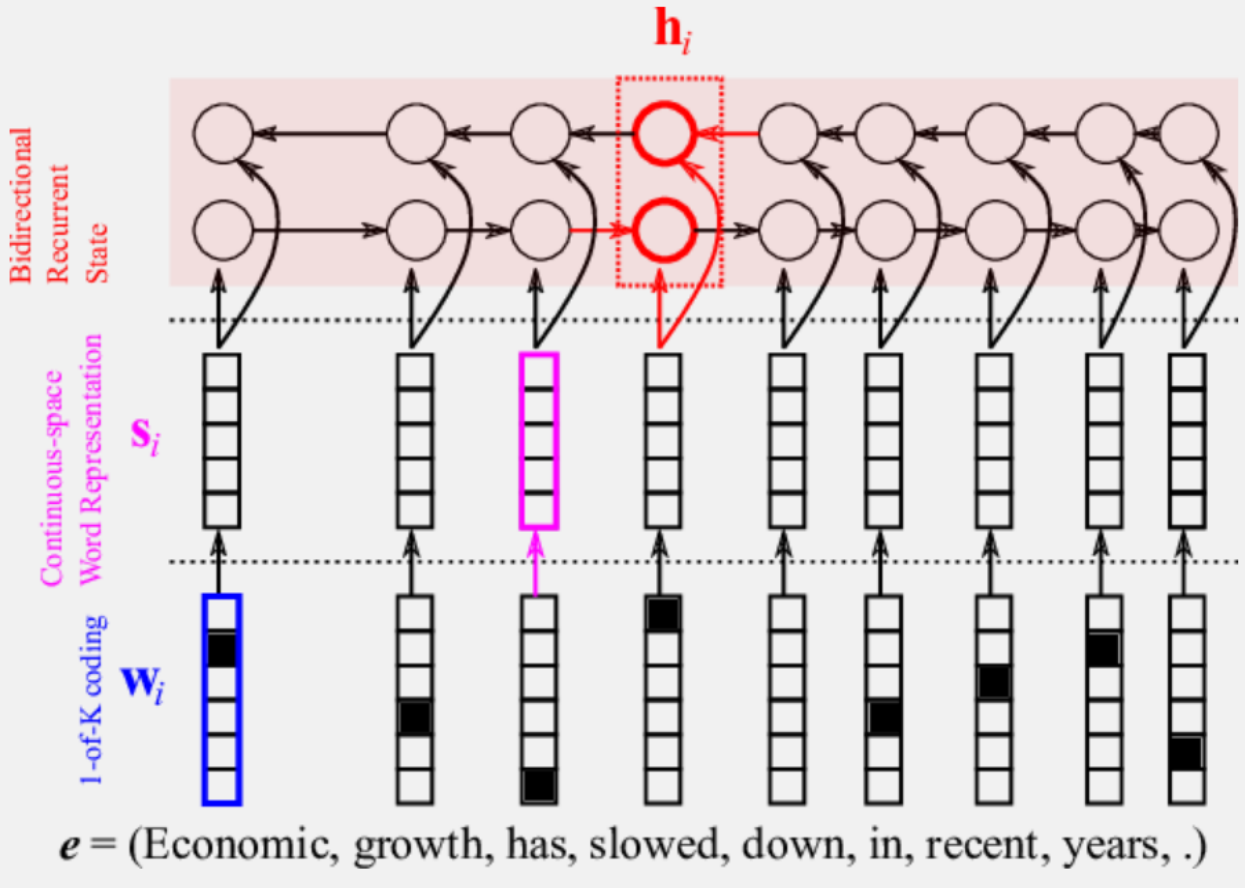


Bidirectional RNNs

A bidirectional recurrent state can capture more dependencies.

Bidirectional RNNs

A bidirectional recurrent state can capture more dependencies.



Greedy decoding

There are various techniques for decoding, the easiest one is known as **greedy decoding**.

Greedy decoding

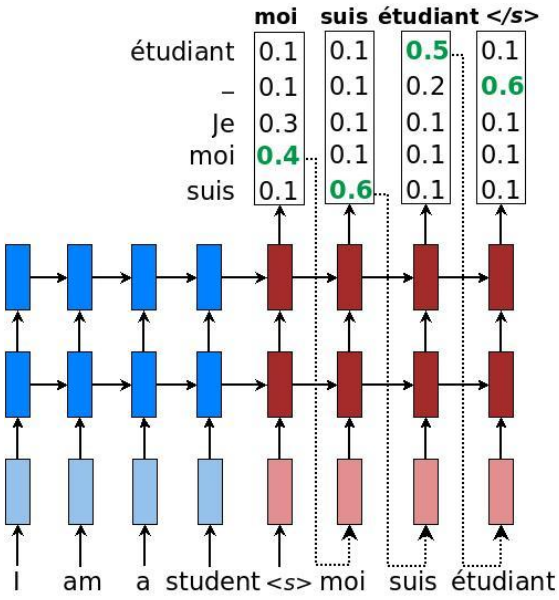
There are various techniques for decoding, the easiest one is known as **greedy decoding**.

Simply choose the word associated with the maximum logit value in the RNN output.

Greedy decoding

There are various techniques for decoding, the easiest one is known as **greedy decoding**.

Simply choose the word associated with the maximum logit value in the RNN output.



Attention-based decoding

Compute attention weights by comparing various words.

Attention-based decoding

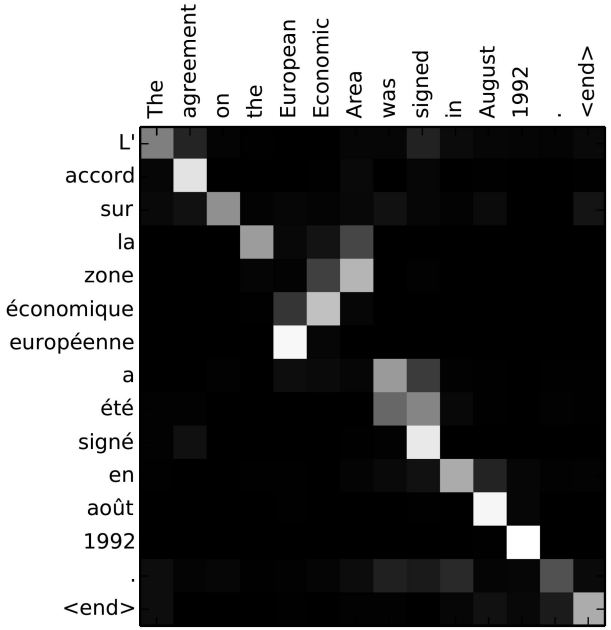
Compute attention weights by comparing various words.

Extends the basic idea of word alignments.

Attention-based decoding

Compute attention weights by comparing various words.

Extends the basic idea of word alignments.

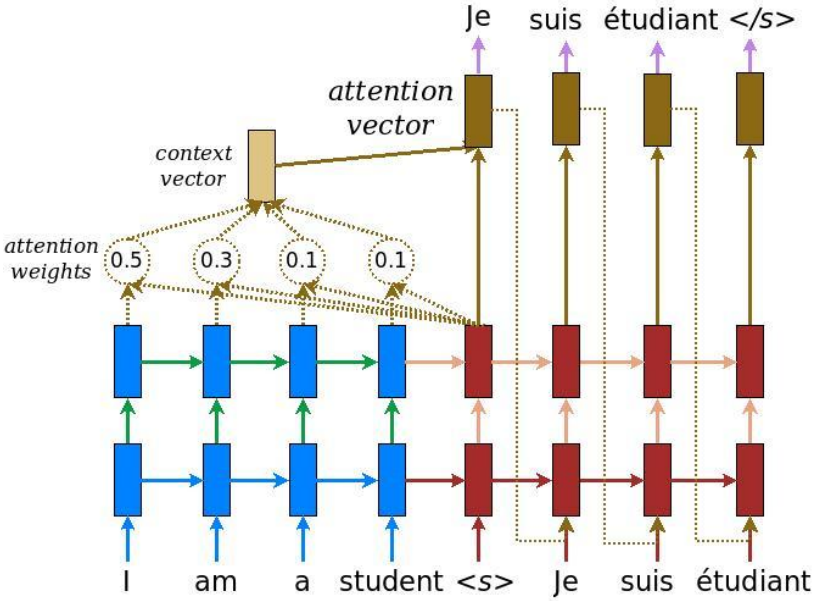


Attention-based decoding

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) \quad \text{[Attention vector]} \quad (3)$$



1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

Text classification

Text classification

The process of assigning tags or categories to text according to its content.

Text classification

Text classification

The process of assigning tags or categories to text according to its content.

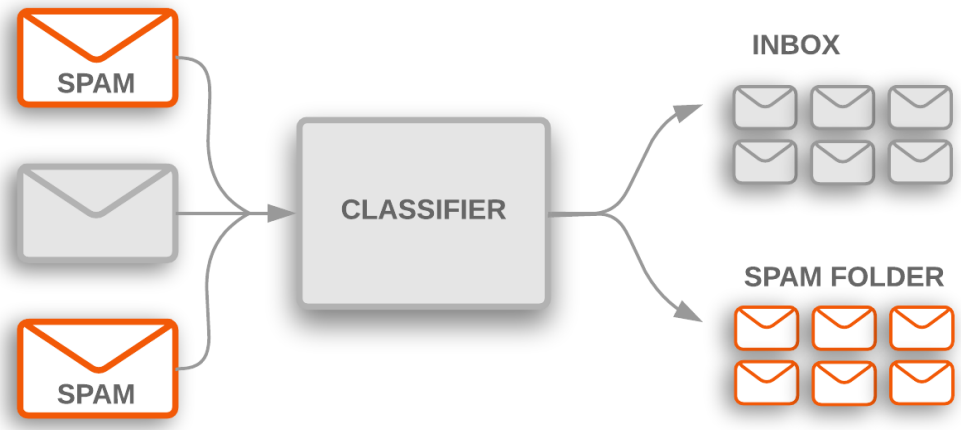
It has many applications, such as author identification, sentiment analysis, fraud detection, etc.

Text classification

Text classification

The process of assigning tags or categories to text according to its content.

It has many applications, such as author identification, sentiment analysis, fraud detection, etc.



Rule-based models for classification

- If we had to build a classifier (ham-spam) from scratch, linguistic rules sound like a good start.
- For example, spam emails tend to include a particular set of words and phrases, which could be our features.

Rule-based models for classification

- If we had to build a classifier (ham-spam) from scratch, linguistic rules sound like a good start.
- For example, spam emails tend to include a particular set of words and phrases, which could be our features.

Rule-based models for classification

- If we had to build a classifier (ham-spam) from scratch, linguistic rules sound like a good start.
- For example, spam emails tend to include a particular set of words and phrases, which could be our features.

Message	Prediction
im donee. come pick me up	Ham
WINNER$$$$ SMS REPLY “WIN”	Spam
whats the matter wit u	Ham
Come to think of it, i never got a spam text messgae before	Ham

Naive Bayes for classification

$$\hat{y} = \arg\max_{y \in Y} \; P(y|x) = \arg\max_{y \in Y} \; \frac{P(x|y)P(y)}{P(x)} \propto \arg\max_{y \in Y} \; P(x|y)P(y)$$

Naive Bayes for classification

$$\hat{y} = \arg\max_{y \in Y} P(y|x) = \arg\max_{y \in Y} \frac{P(x|y)P(y)}{P(x)} \propto \arg\max_{y \in Y} P(x|y)P(y)$$

D1: “send us your password” spam
D2: “send us your review” ham
D3: “review your password” ham
D4: “review us” spam
D5: “send your password” spam
D6: “send us your account” spam

new email: “review us now”

P (spam) = 4/6 P (ham) = 2/6

spam	ham	
2/4	1/2	password
1/4	2/2	review
3/4	1/2	send
3/4	1/2	us
3/4	1/2	your
1/4	0/2	account

$$P(\text{review us}|\text{spam}) = P(0,1,0,1,0,0|\text{spam}) = (1-\frac{2}{4})(\frac{1}{4})(1-\frac{3}{4})(\frac{3}{4})(1-\frac{3}{4})(1-\frac{1}{4})$$

$$P(\text{review us}|\text{ham}) = P(0,1,0,1,0,0|\text{ham}) = (1-\frac{1}{2})(\frac{2}{2})(1-\frac{1}{2})(\frac{1}{2})(1-\frac{1}{2})(1-\frac{0}{2})$$

$$P(\text{ham}|\text{review us}) = \frac{0.0625 \times 2/6}{0.0625 \times 2/6 + 0.0044 \times 4/6} = 0.87$$ (note identical example)

Decision trees for classification

A mathematically sound way to derive a set of rules.

Decision trees for classification

A mathematically sound way to derive a set of rules.

Predict if John will play tennis

- Hard to guess
- Divide & conquer:
  - split into subsets
  - are they pure? (all yes or all no)
  - if yes: stop
  - if not: repeat
- See which subset new data falls into

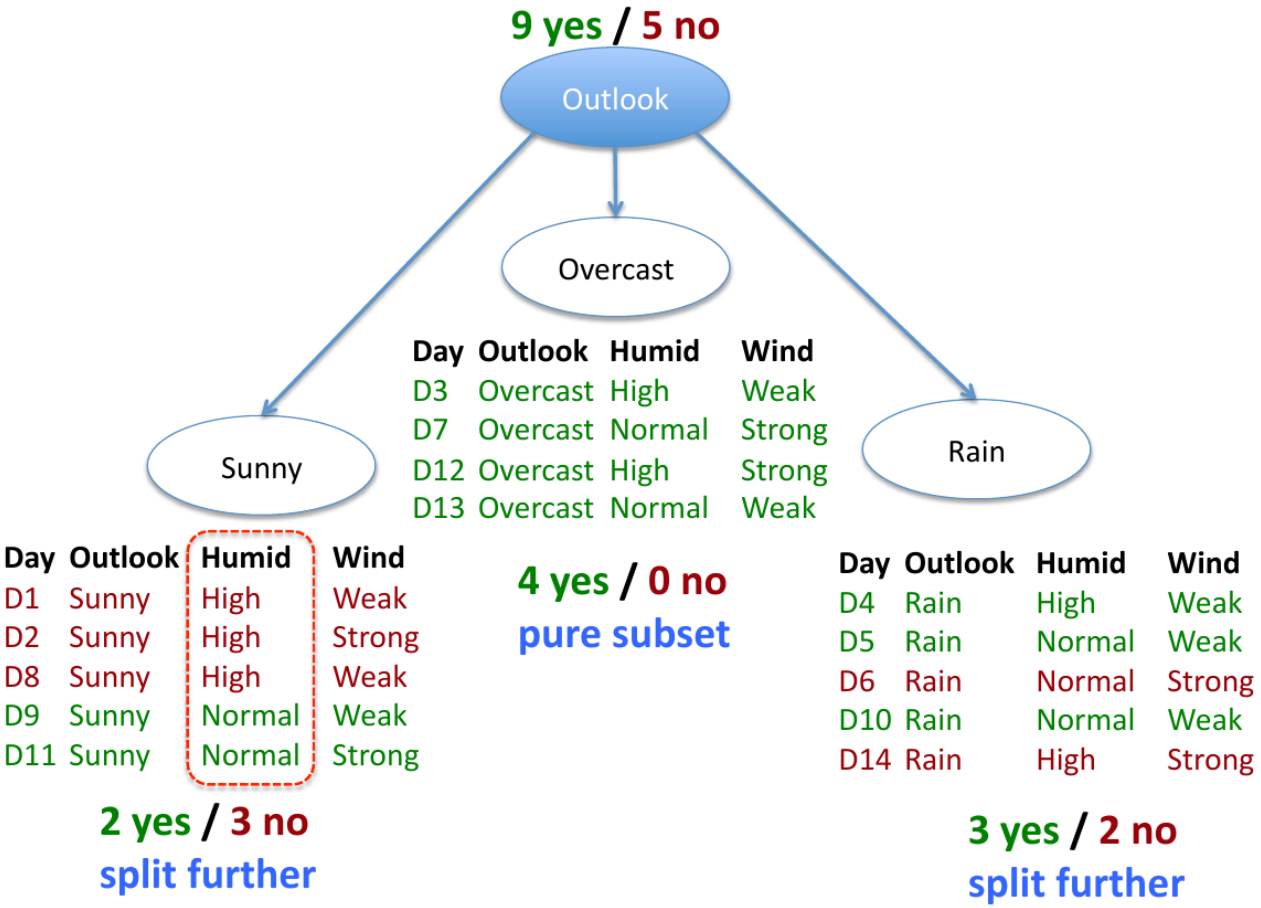
Training examples: **9 yes / 5 no**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

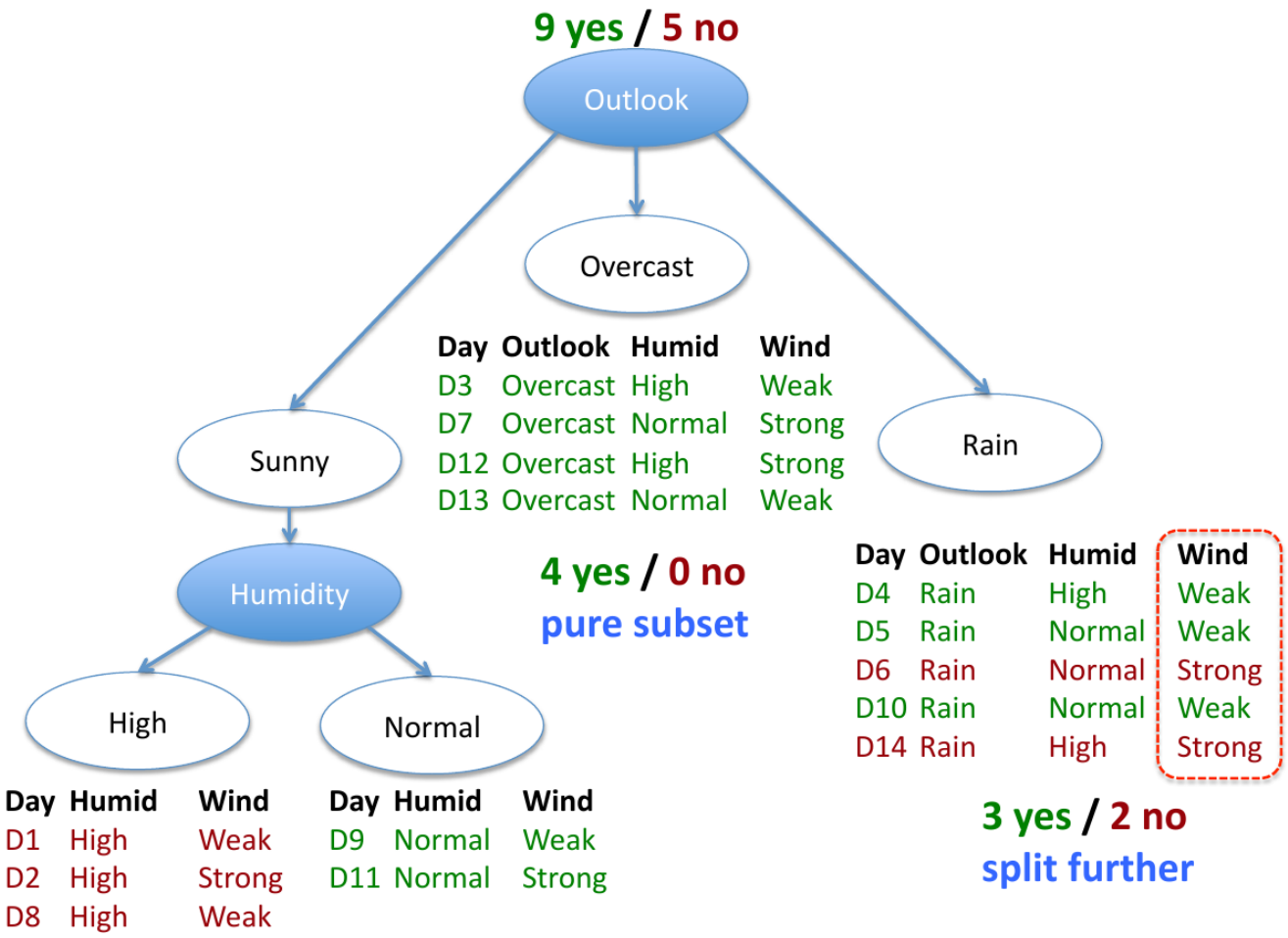
New data:

D15	Rain	High	Weak	?

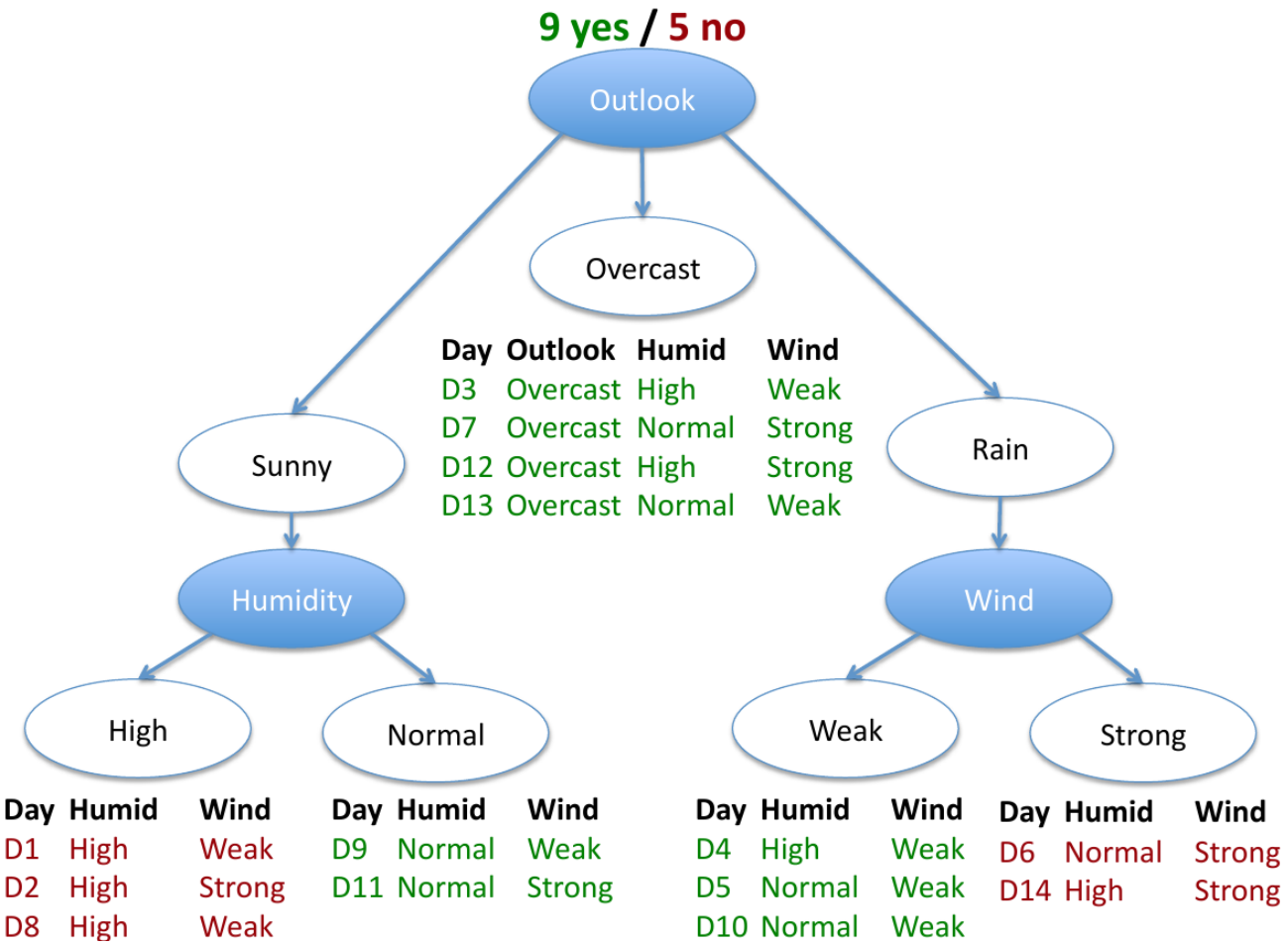
Decision trees for classification



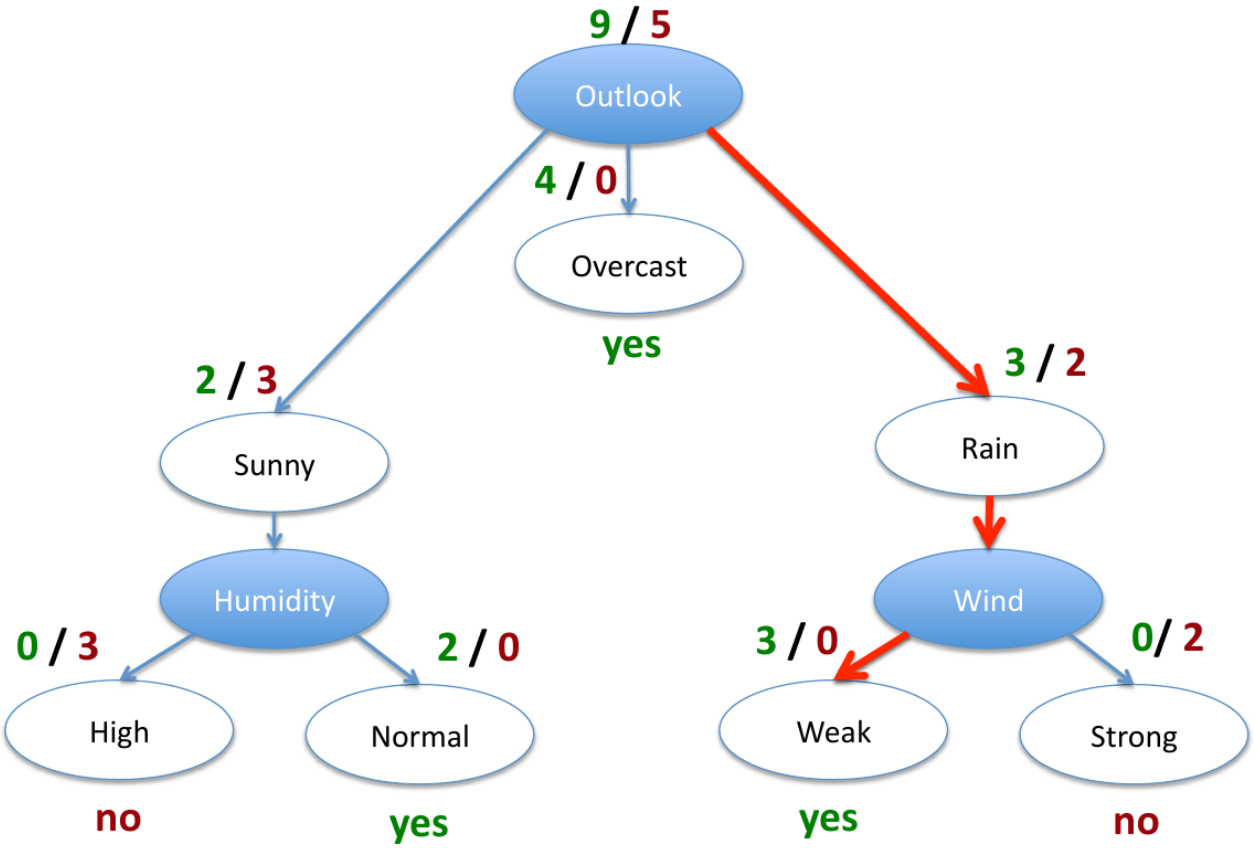
Decision trees for classification



Decision trees for classification



Decision trees for classification



New data:

Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	→ Yes

Which attribute to split on?

We want a split that delivers many items in pure sets. **Information gain** measures the expected drop of entropy after a split.

Which attribute to split on?

We want a split that delivers many items in pure sets. **Information gain** measures the expected drop of entropy after a split.

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

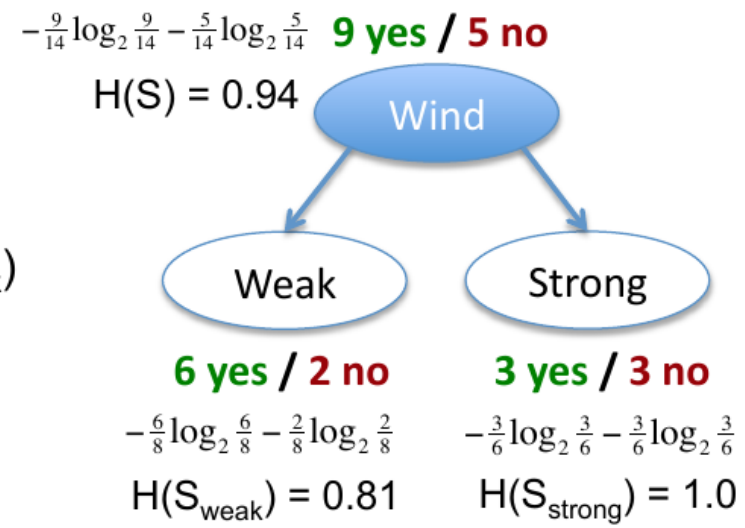
V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

Which attribute to split on?

We want a split that delivers many items in pure sets. **Information gain** measures the expected drop of entropy after a split.

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$



Gain (S, Wind)
$= H(S) - {}^{8}/_{14}\, H(S_{weak}) - {}^{6}/_{14}\, H(S_{weak})$
$= 0.94 - {}^{8}/_{14} * 0.81 - {}^{6}/_{14} * 1.0$
$= 0.049$

Problems with decision trees

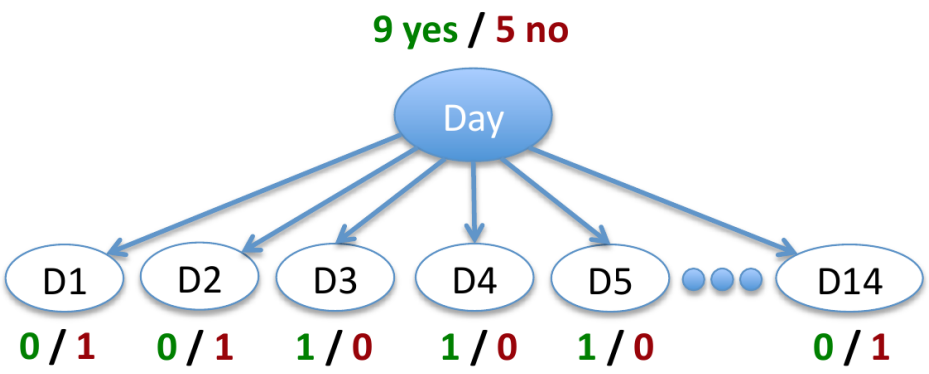
- The training examples can always be classified perfectly, which leads to **overfitting**.
- Information gain is biased towards attributes with many values.

Problems with decision trees

- The training examples can always be classified perfectly, which leads to **overfitting**.
- Information gain is biased towards attributes with many values.

Problems with decision trees

- The training examples can always be classified perfectly, which leads to **overfitting**.
- Information gain is biased towards attributes with many values.



Random decision forest

Random decision forest

Ensemble model that grows and uses K different decision trees.

- Each of the trees works on a random subset of the data and attributes.
- New data points are classified using each of the K trees. The final class is decided by majority voting.

Random decision forest

Random decision forest

Ensemble model that grows and uses K different decision trees.

- Each of the trees works on a random subset of the data and attributes.
- New data points are classified using each of the K trees. The final class is decided by majority voting.

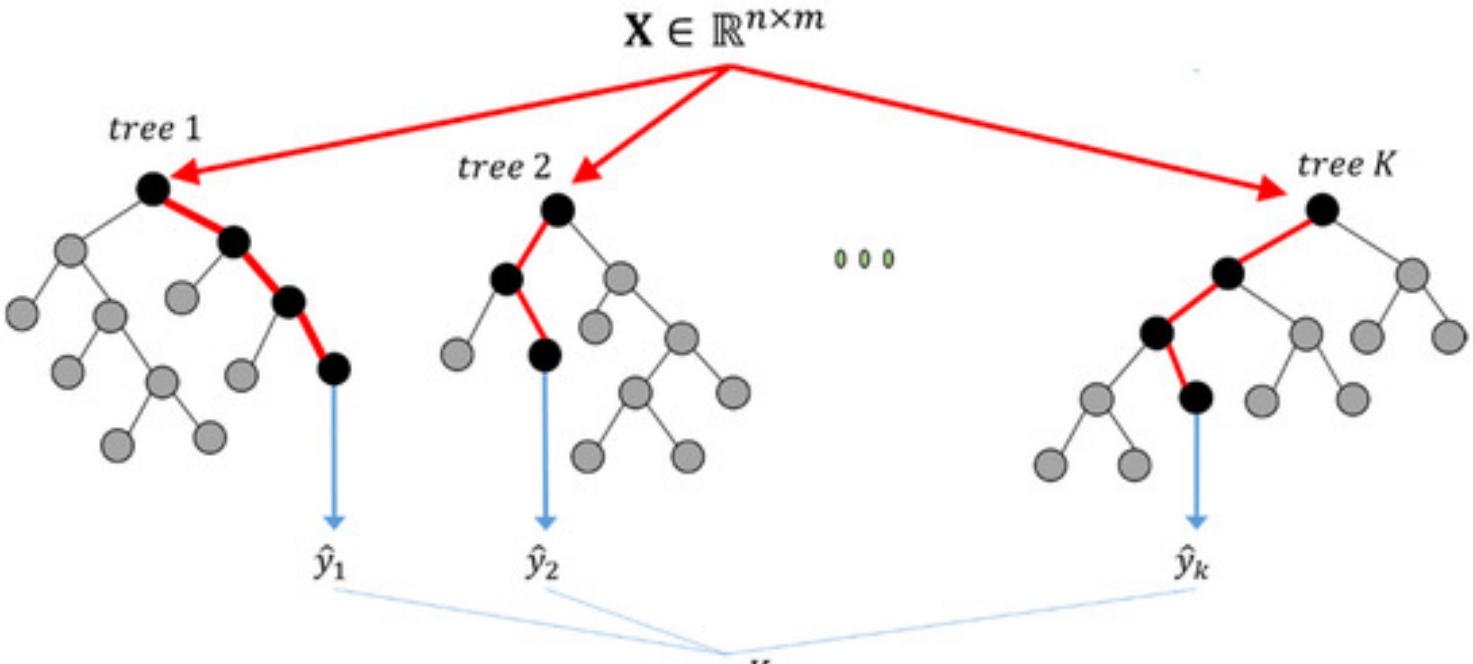
Random decision forest

Random decision forest

Ensemble model that grows and uses K different decision trees.

- Each of the trees works on a random subset of the data and attributes.
- New data points are classified using each of the K trees. The final class is decided by majority voting.

Random decision forest



$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} \hat{y}_k$$

Support Vector Machine

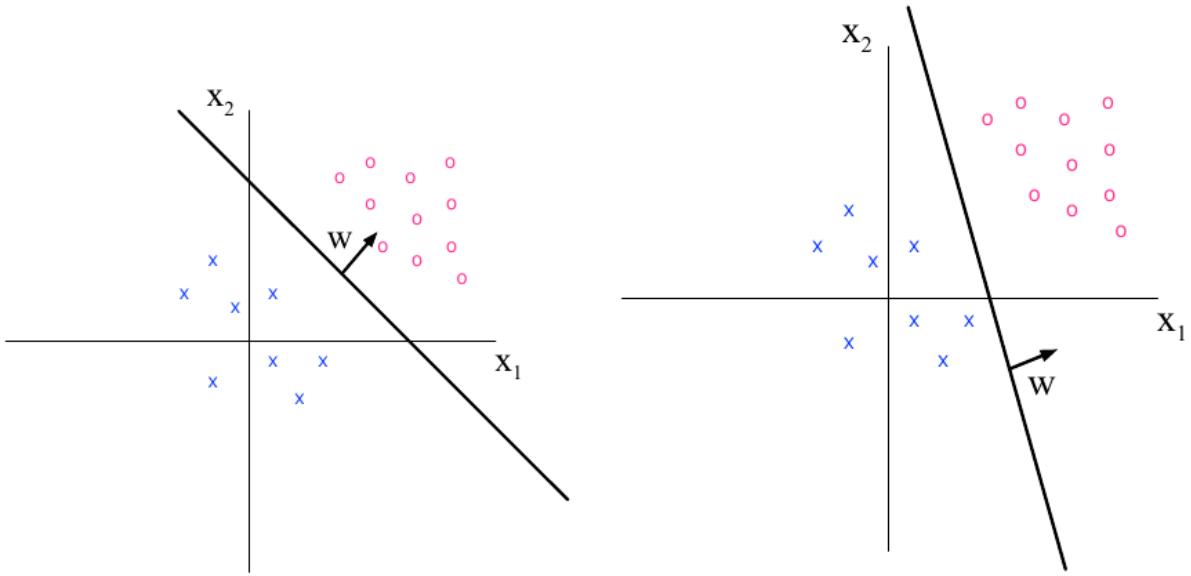
- **Support Vector Machines (SVM)** are one of the most effective and used classification algorithms.
- They are focused on finding a separating hyperplane for the input data.

Support Vector Machine

- **Support Vector Machines (SVM)** are one of the most effective and used classification algorithms.
- They are focused on finding a separating hyperplane for the input data.

Support Vector Machine

- **Support Vector Machines (SVM)** are one of the most effective and used classification algorithms.
- They are focused on finding a separating hyperplane for the input data.

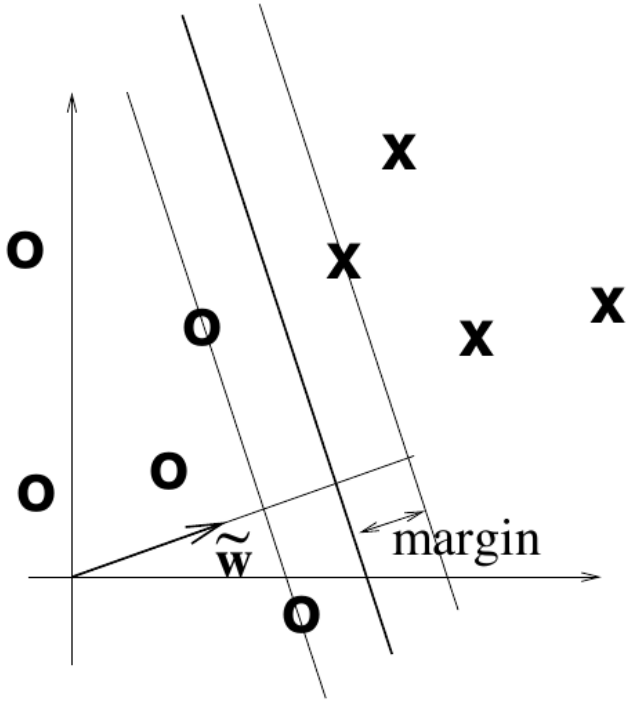


Maximizing the margin

The **margin** is the distance between the decision boundary and the closest training point.

Maximizing the margin

The **margin** is the distance between the decision boundary and the closest training point.



Maximizing the margin

We would like to get the hyperplane that maximizes the margin.

Maximizing the margin

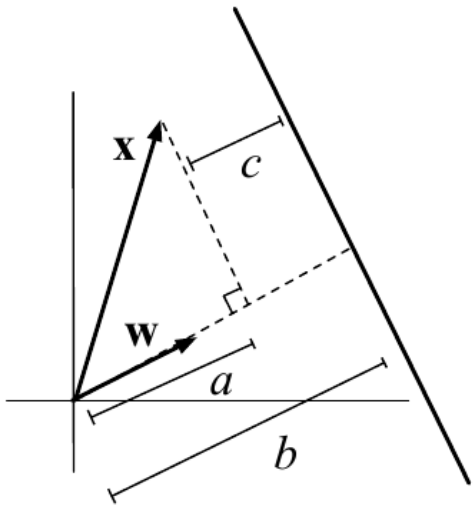
We would like to get the hyperplane that maximizes the margin.

The math required is beyond this talk, but essentially...

Maximizing the margin

We would like to get the hyperplane that maximizes the margin.

The math required is beyond this talk, but essentially...



$$
\begin{aligned}
\max_{\mathbf{w}} \quad & 1/||\mathbf{w}|| \\
\text{subject to} \quad & \mathbf{w}^\top \mathbf{x}_i + w_0 \geq 1 && \text{for all } i \text{ with } y_i = 1 \\
& \mathbf{w}^\top \mathbf{x}_i + w_0 \leq -1 && \text{for all } i \text{ with } y_i = -1
\end{aligned}
$$

Transfer learning

Transfer learning

A machine learning technique that leverages the knowledge acquired on a certain task in order to solve a separate related task.

Transfer learning

Transfer learning

A machine learning technique that leverages the knowledge acquired on a certain task in order to solve a separate related task.

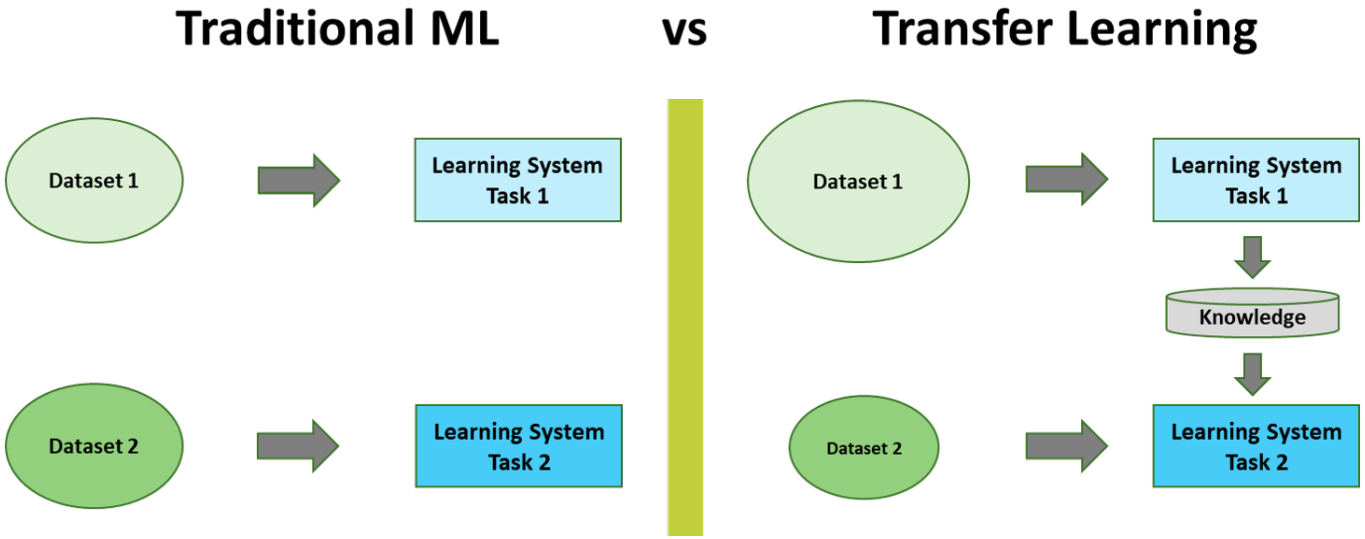
ULMFiT is a model which takes advantage of transfer learning. The approach is reminiscent to what we saw with **word2vec**

Transfer learning

Transfer learning

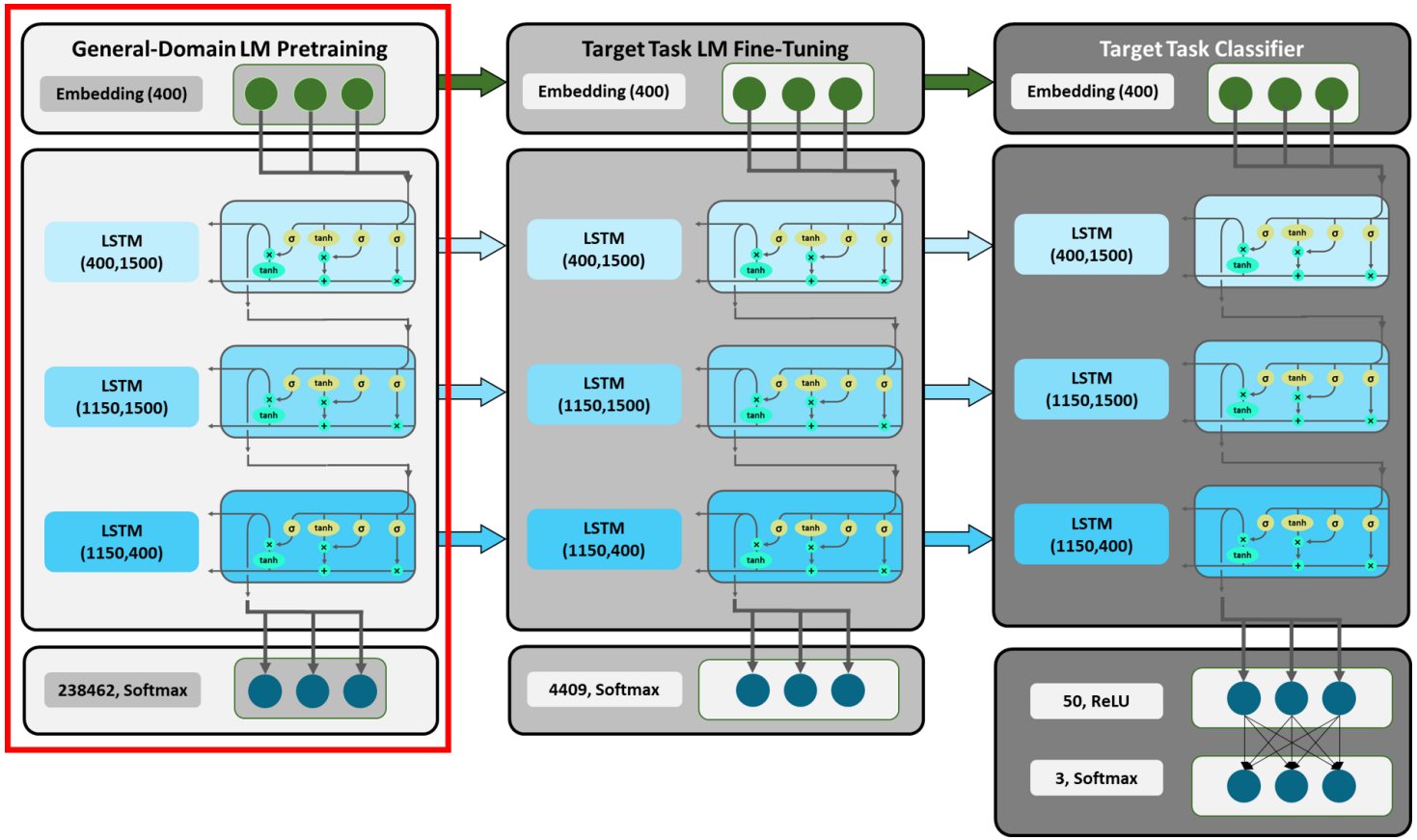
A machine learning technique that leverages the knowledge acquired on a certain task in order to solve a separate related task.

ULMFiT is a model which takes advantage of transfer learning. The approach is reminiscent to what we saw with **word2vec**



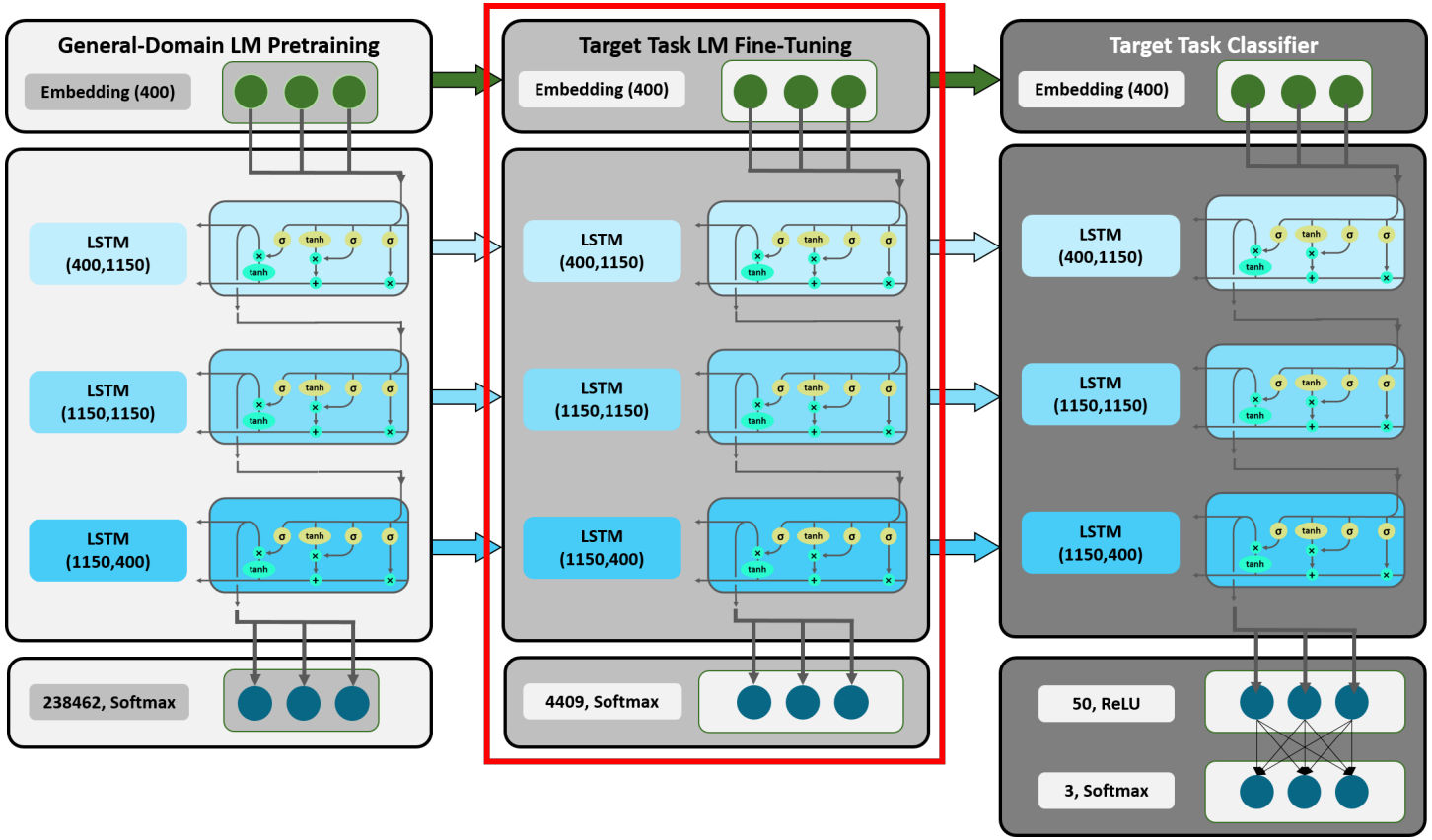
Overview of ULMFiT

General-Domain LM Pretraining is where the model learns the general language features (e.g. subject-verb-object order).



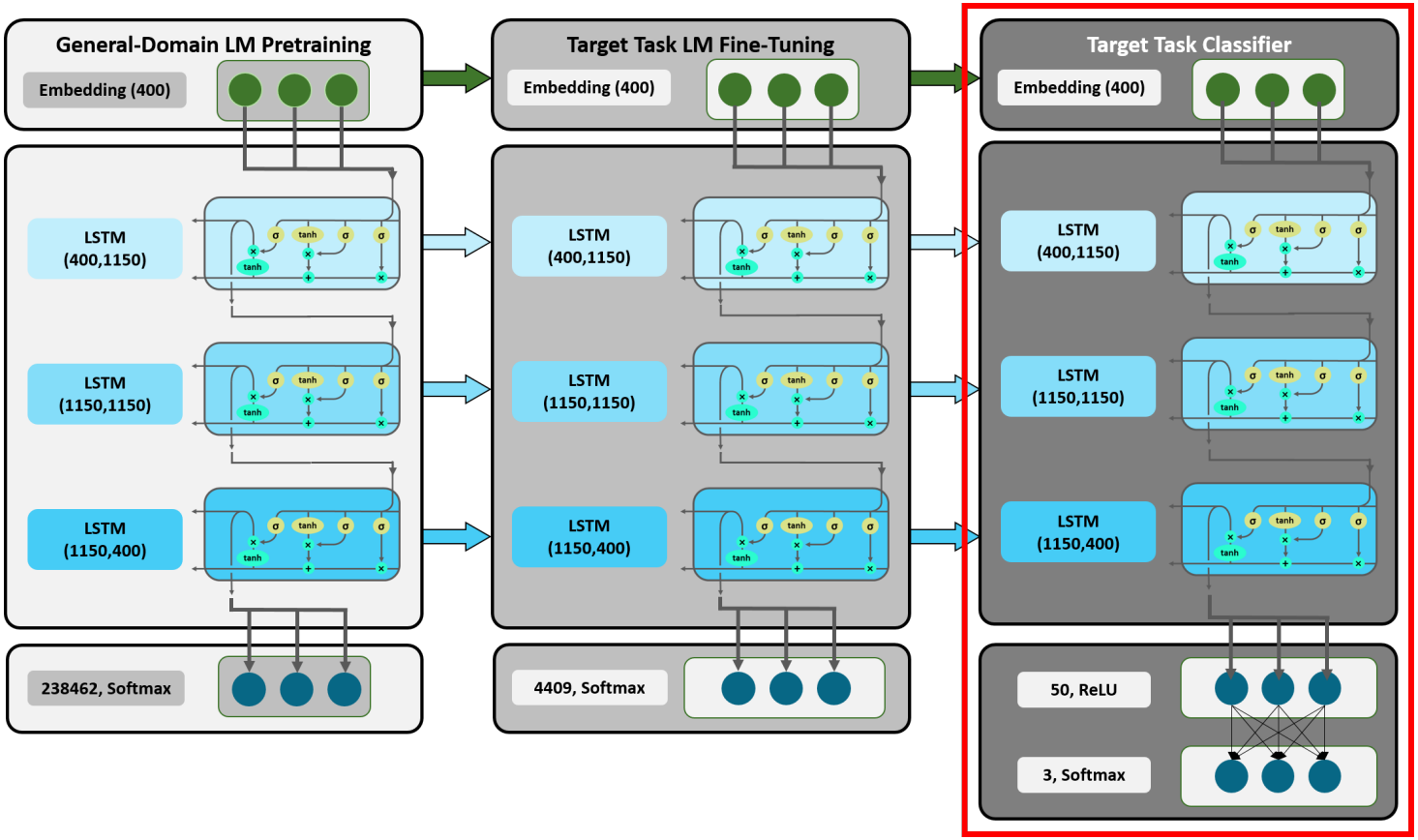
Overview of ULMFiT

Target Task LM Fine-Tuning is a stage where the model learns the in-domain features of the language specific to a certain task.



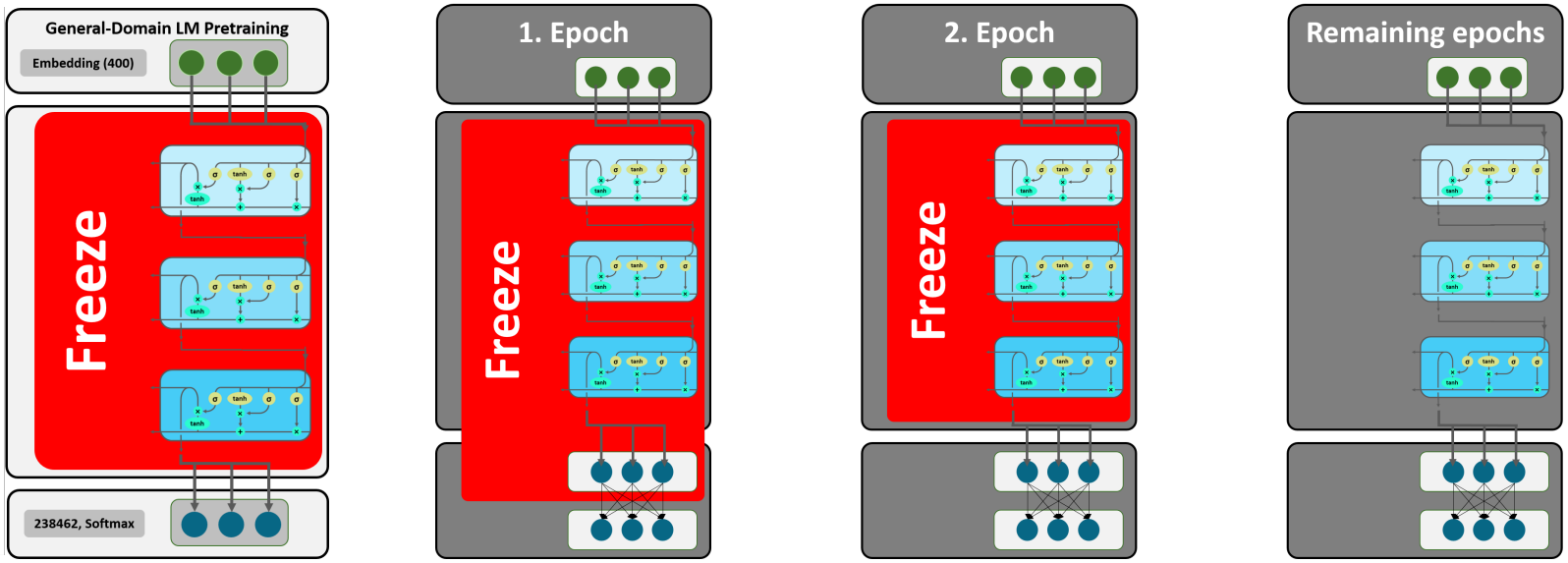
Overview of ULMFiT

A **Target Task Clasifier** is built by expanding the fine-tuned language model to a probability distribution over labels.



Gradual Unfreezing

Freezing is a technique that can be used in order to only train certain layers of the network sequentially.



1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

Transformer models

- **BERT** is a model with the same operating principle as ULMFiT.
- It uses the **transformer** architecture, instead of an explicit RNN.
- Transformers use an encoder-decoder model together with **attention** to mutate sequences.

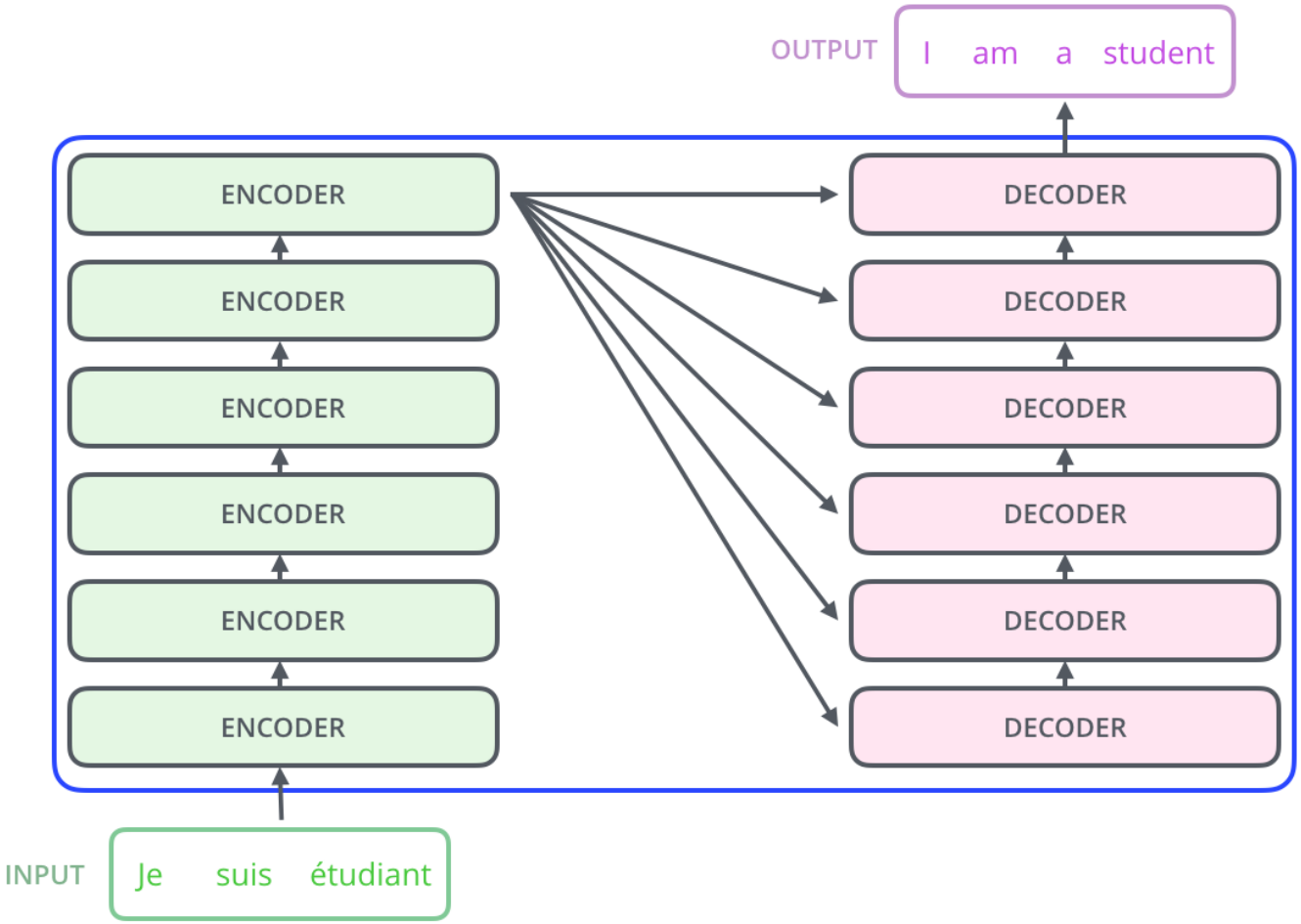
Transformer models

- **BERT** is a model with the same operating principle as ULMFiT.
- It uses the **transformer** architecture, instead of an explicit RNN.
- Transformers use an encoder-decoder model together with **attention** to mutate sequences.

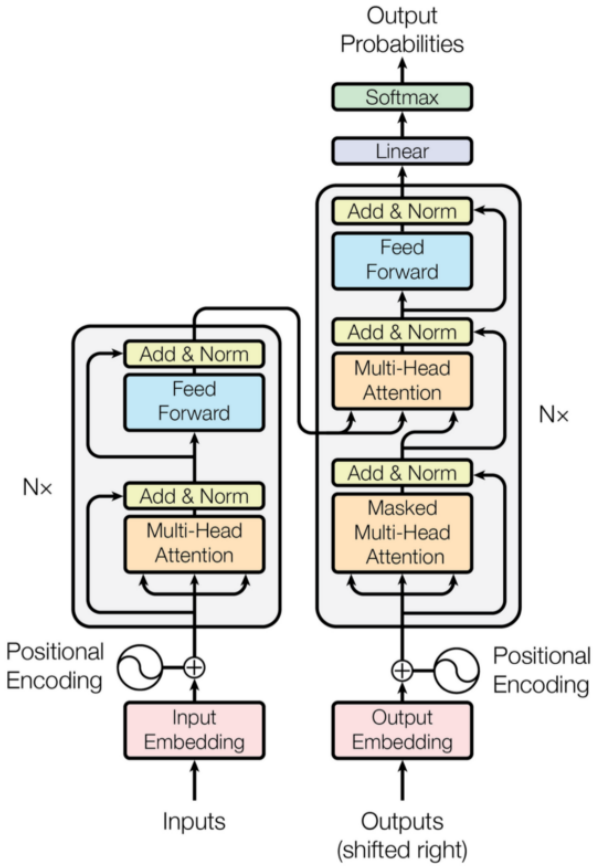
Transformer models

- **BERT** is a model with the same operating principle as ULMFiT.
- It uses the **transformer** architecture, instead of an explicit RNN.
- Transformers use an encoder-decoder model together with **attention** to mutate sequences.

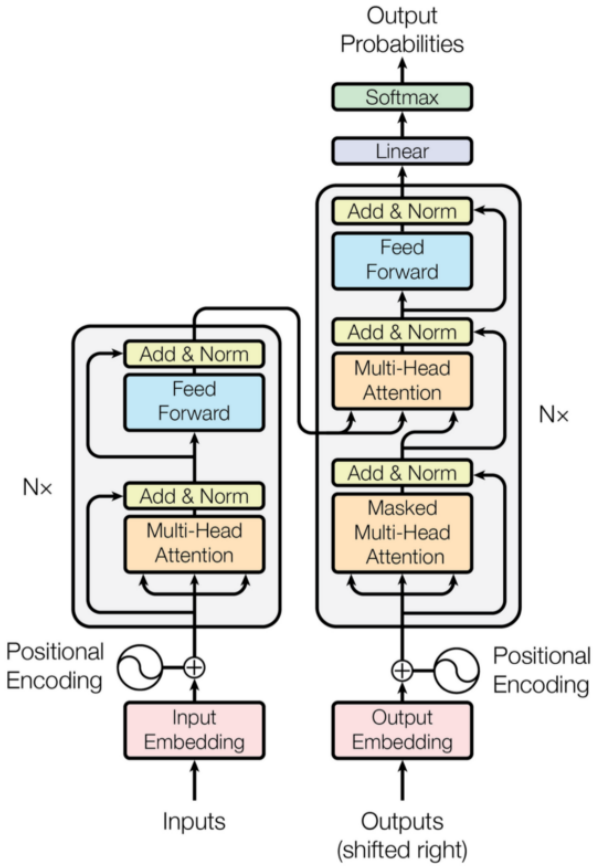
Transformer models



Transformer models



Transformer models



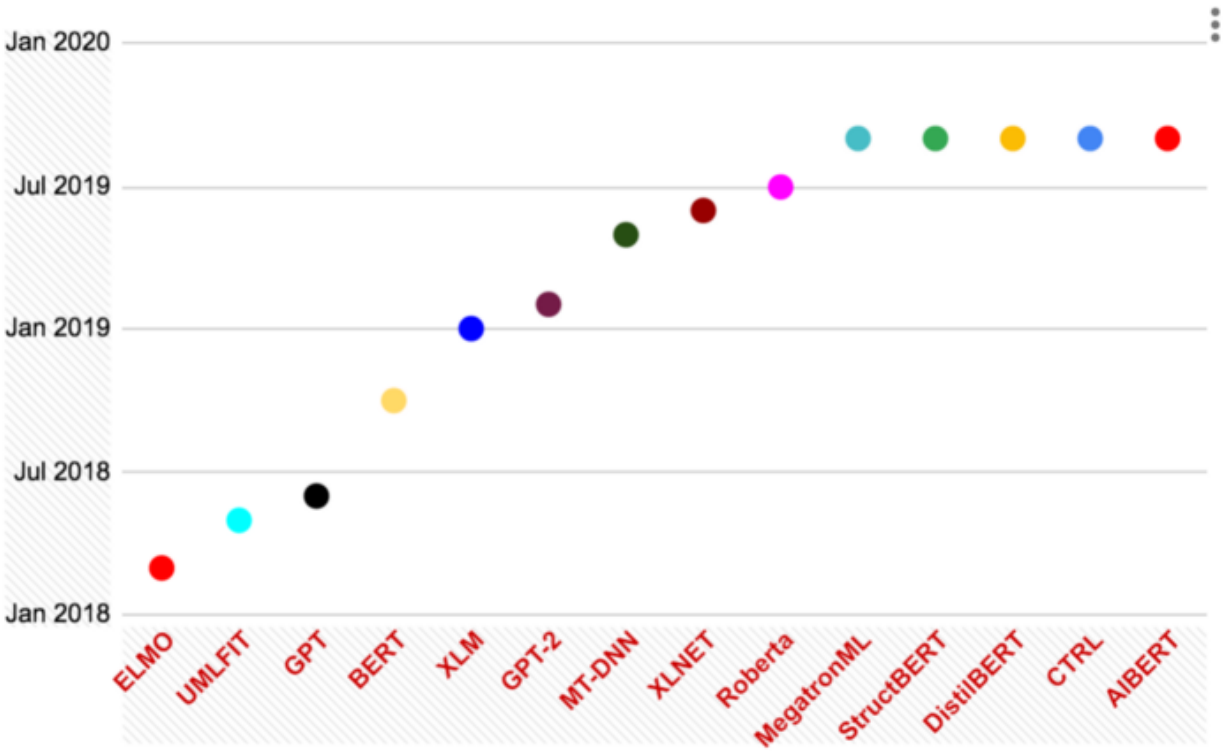
In BERT, only the encoder is needed.

BERT summary

BERT is one of the most popular models in state-of-the-art architectures and has many variations.

BERT summary

BERT is one of the most popular models in state-of-the-art architectures and has many variations.



BERT summary

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step

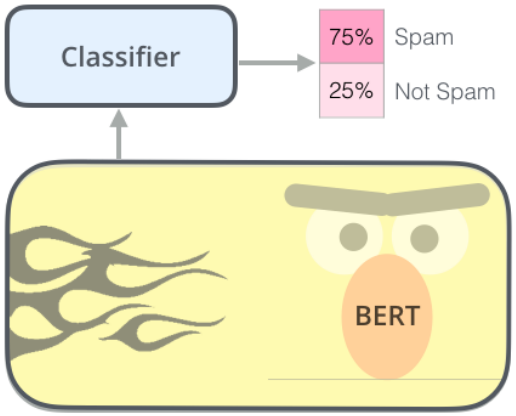
Model:

Dataset:

Objective: Predict the masked word (langauge modeling)

2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step



Model: (pre-trained in step #1)

Dataset:

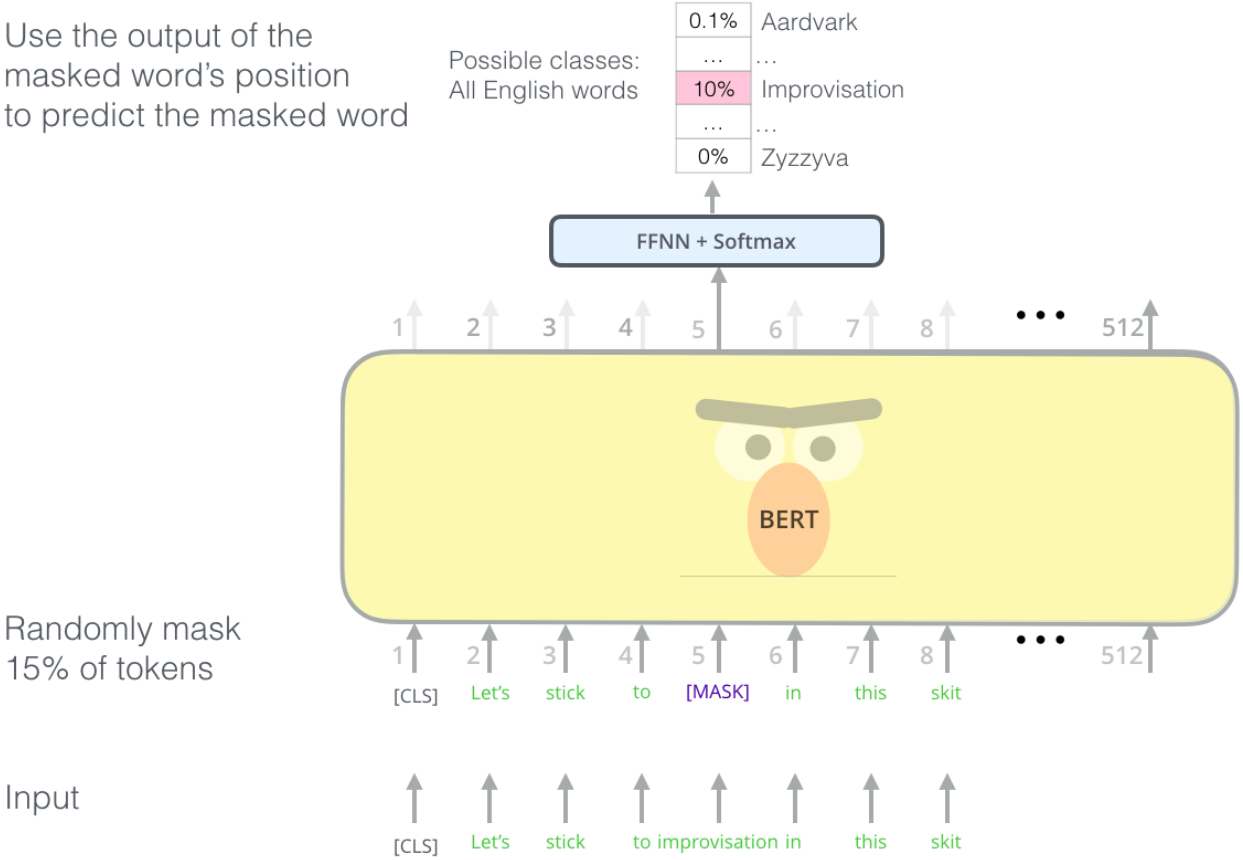
Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

BERT training

BERT is mainly trained on a task called **Masked Language Modelling**, which attempts to predict masked words.

BERT training

BERT is mainly trained on a task called **Masked Language Modelling**, which attempts to predict masked words.

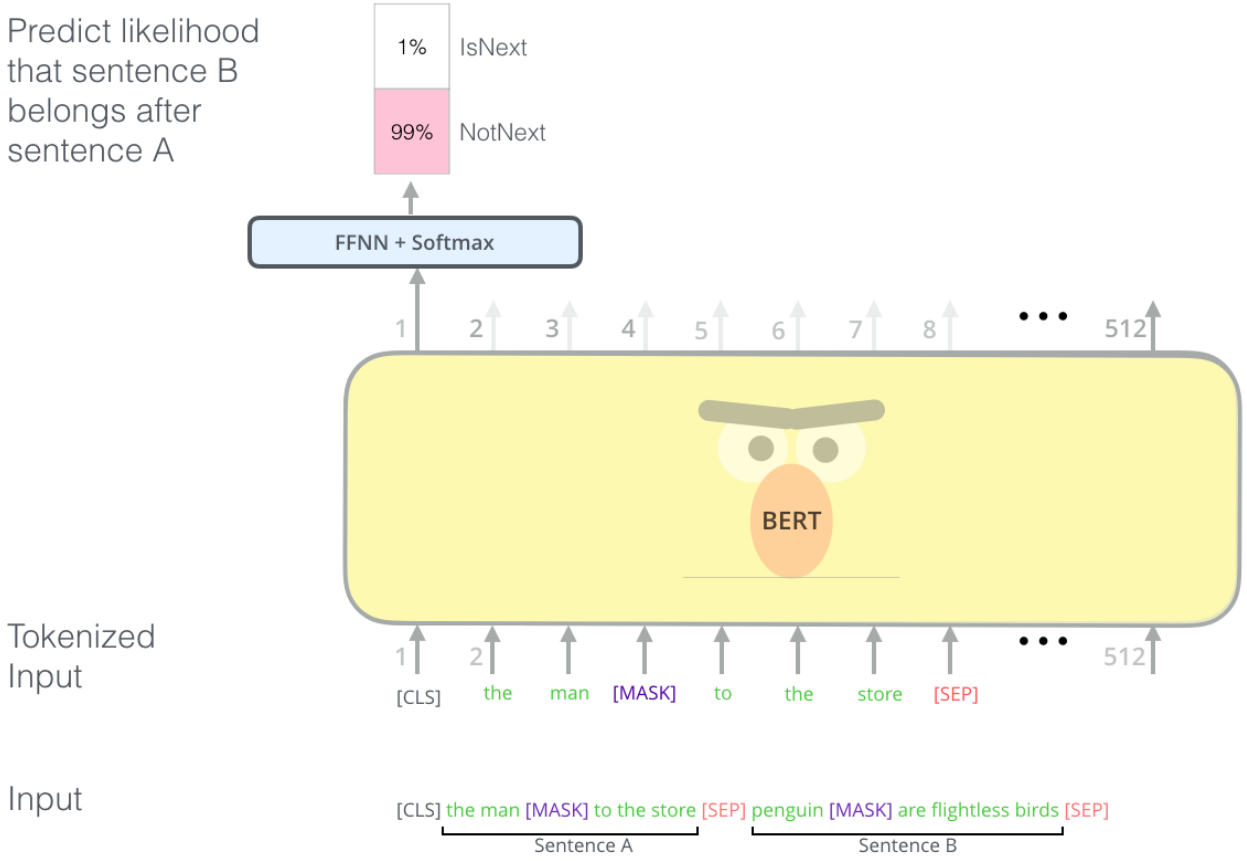


BERT training

In addition, BERT is also trained on another task called **Next Sentence Prediction**.

BERT training

In addition, BERT is also trained on another task called **Next Sentence Prediction**.

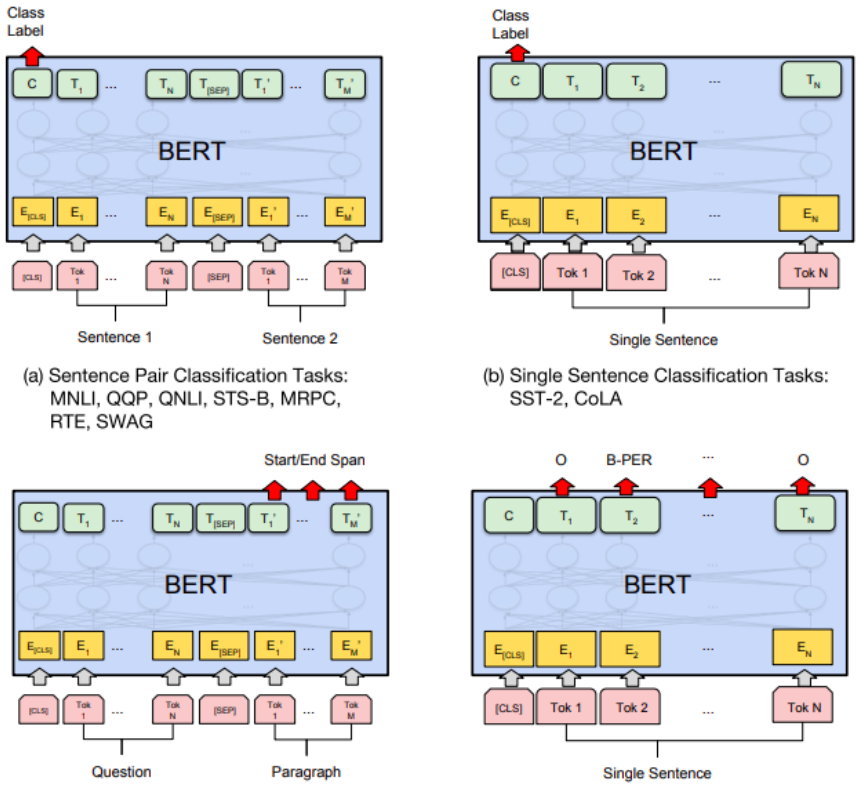


Using BERT

Multiple tasks achieve state-of-the-art performance when solved using an appropriately trained BERT model.

Using BERT

Multiple tasks achieve state-of-the-art performance when solved using an appropriately trained BERT model.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT results

#	Model	SST-2	QQP	MNLI-m	MNLI-mm
		Acc	F_1/Acc	Acc	Acc
1	$BERT_{LARGE}$ (Devlin et al., 2018)	94.9	72.1/89.3	86.7	85.9
2	$BERT_{BASE}$ (Devlin et al., 2018)	93.5	71.2/89.2	84.6	83.4
3	OpenAI GPT (Radford et al., 2018)	91.3	70.3/88.5	82.1	81.4
4	BERT ELMo baseline (Devlin et al., 2018)	90.4	64.8/84.7	76.4	76.1
5	GLUE ELMo baseline (Wang et al., 2018)	90.4	63.1/84.3	74.1	74.5
6	Distilled $BiLSTM_{SOFT}$	**90.7**	**68.2/88.1**	**73.0**	**72.6**
7	BiLSTM (our implementation)	86.7	63.7/86.2	68.7	68.3
8	BiLSTM (reported by GLUE)	85.9	61.4/81.7	70.3	70.8
9	BiLSTM (reported by other papers)	$87.6^{\dagger}$	– /$82.6^{\ddagger}$	66.9^{*}	66.9^{*}

Table 1: Test results on different datasets. The BiLSTM results reported by other papers are drawn from Zhou et al. (2016),† Wang et al. (2017),‡ and Williams et al. (2017).* All of our test results are obtained from the GLUE benchmark website.

Problems with BERT

- BERT is a really big model, costly to train and use.
- There is a recent trend towards developing smaller models.
- ELECTRA is a new model which claims to be simpler than BERT but with similar performance.

Problems with BERT

- BERT is a really big model, costly to train and use.
- There is a recent trend towards developing smaller models.
- ELECTRA is a new model which claims to be simpler than BERT but with similar performance.

Problems with BERT

- BERT is a really big model, costly to train and use.
- There is a recent trend towards developing smaller models.
- ELECTRA is a new model which claims to be simpler than BERT but with similar performance.

ELECTRA

- The **ELECTRA** model is inspired by GANs.
- The model is trained on a **discrimination** task, rather than a generation one.
- This task is called replaced token detection.

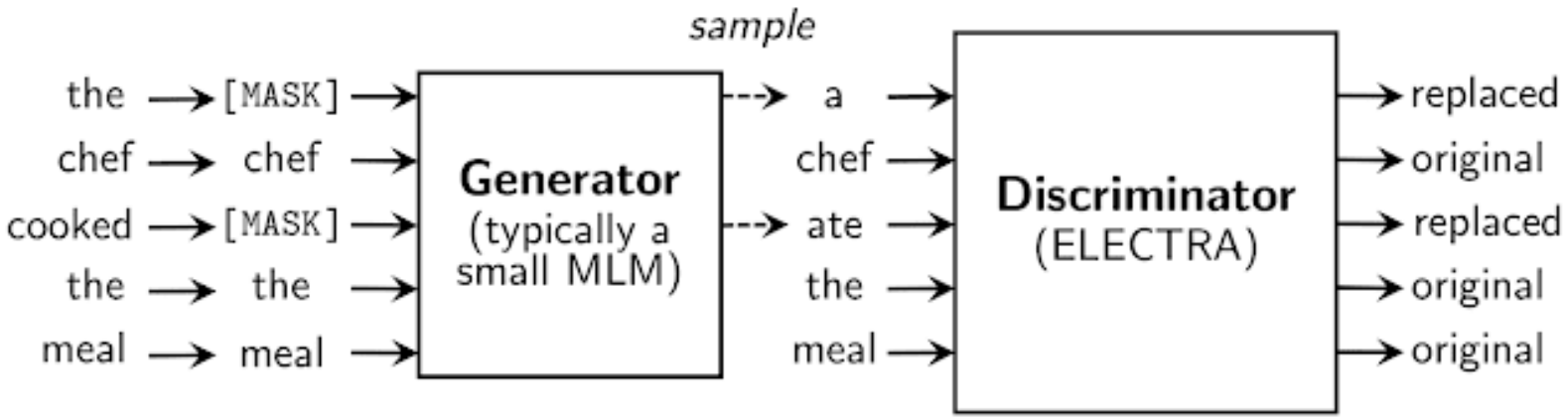
ELECTRA

- The **ELECTRA** model is inspired by GANs.
- The model is trained on a **discrimination** task, rather than a generation one.
- This task is called replaced token detection.

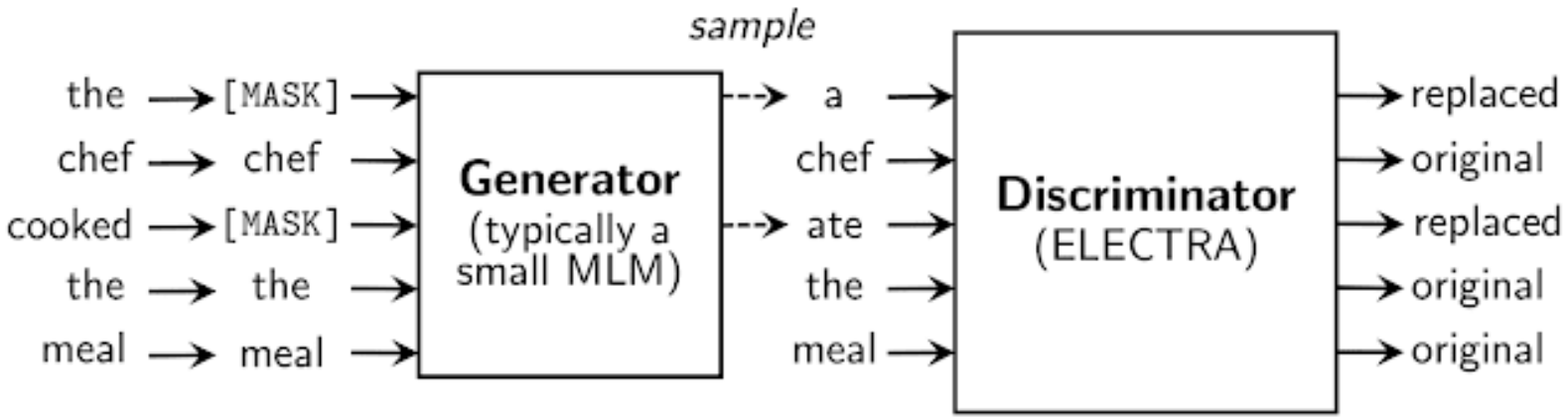
ELECTRA

- The **ELECTRA** model is inspired by GANs.
- The model is trained on a **discrimination** task, rather than a generation one.
- This task is called replaced token detection.

ELECTRA

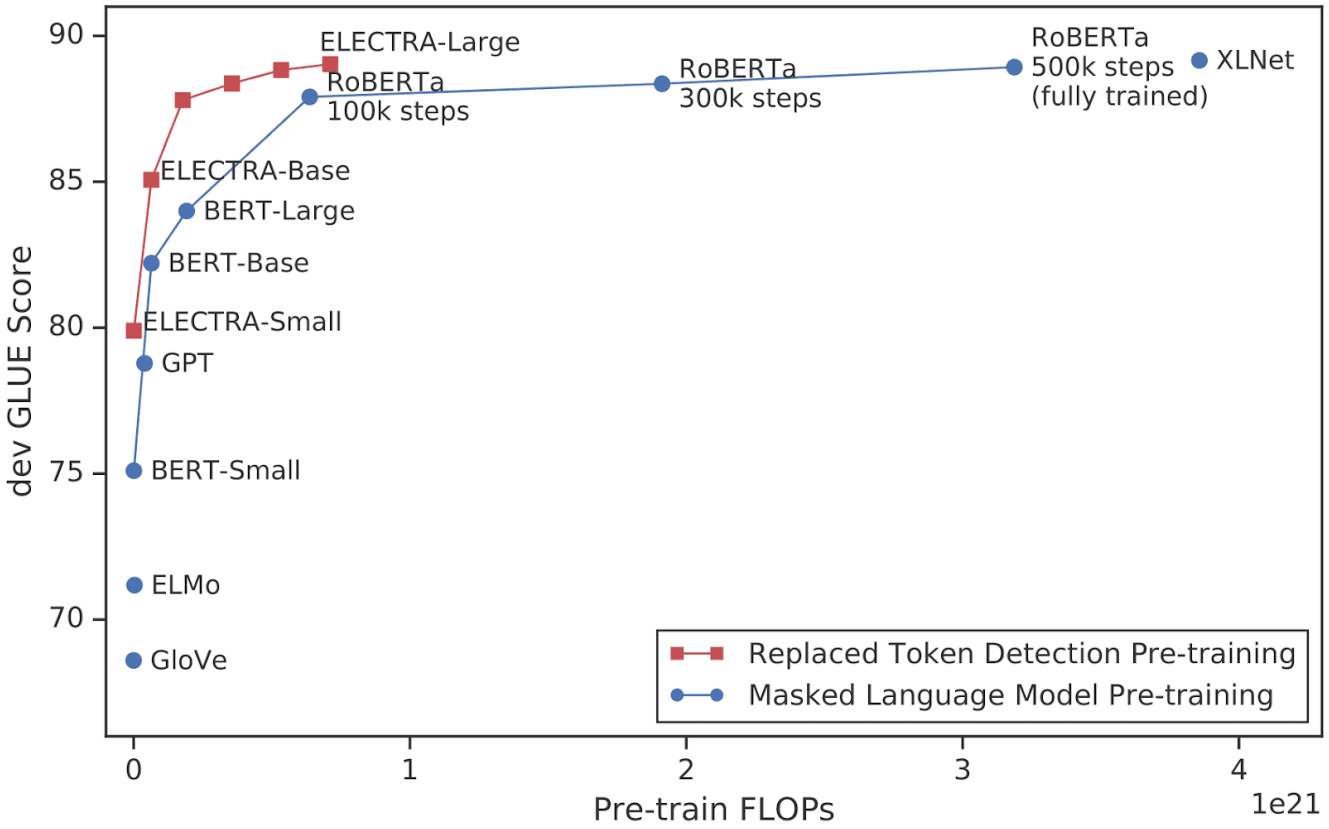

sample
the
chef
cooked
the
meal
[MASK]
chef
[MASK]
the
meal
Generator
(typically a
small MLM)
a
chef
ate
the
meal
Discriminator
(ELECTRA)
replaced
original
replaced
original
original

ELECTRA



ELECTRA uses the discriminator block to extract features.

ELECTRA results



1 Introduction to language technologies

2 How to turn words into features

3 How to translate between languages

4 How to classify texts

5 Transformers for natural language

6 Conclusion

Take home thoughts

- NLP is a vast domain with many exciting applications.
- We have come a long way... choose the right tool for the job.
- There is a long way ahead... find ways in which linguistics and machine learning can work together.
- Read a lot. Luck is what happens when preparation meets opportunity.

Take home thoughts

- NLP is a vast domain with many exciting applications.
- We have come a long way... choose the right tool for the job.
- There is a long way ahead... find ways in which linguistics and machine learning can work together.
- Read a lot. Luck is what happens when preparation meets opportunity.

Take home thoughts

- NLP is a vast domain with many exciting applications.
- We have come a long way... choose the right tool for the job.
- There is a long way ahead... find ways in which linguistics and machine learning can work together.
- Read a lot. Luck is what happens when preparation meets opportunity.

Take home thoughts

- NLP is a vast domain with many exciting applications.
- We have come a long way... choose the right tool for the job.
- There is a long way ahead... find ways in which linguistics and machine learning can work together.
- Read a lot. Luck is what happens when preparation meets opportunity.

Natural Language Processing (NLP)
State of the art models - Technical class

Joan Ginés i Ametllé

AI Researcher

23rd July 2020

cinnamon AI