Quebec
Artificial
Intelligence
Institute

Mila

Natural Language Processing

Mirko Bronzi
Applied Research Scientist, Mila
mirko.bronzi@mila.quebec

Plan

- Natural Language Processing
- Words and Semantics
- Classical Approaches
- Word Embeddings
- Contextualized Word Embeddings
- BERT
- Summary

Plan

- **Natural Language Processing**
- Words and Semantics
- Classical Approaches
- Word Embeddings
- Contextualized Word Embeddings
- BERT
- Summary

Natural Language Processing

- “Natural Language Processing (NLP) is a subfield of computer science, information engineering, and artificial intelligence concerned with the **interactions between computers and human (natural) languages**, in particular how to program computers to process and analyze large amounts of natural language data.”

- There are many NLP tasks - in the next slides we will look at some of them.

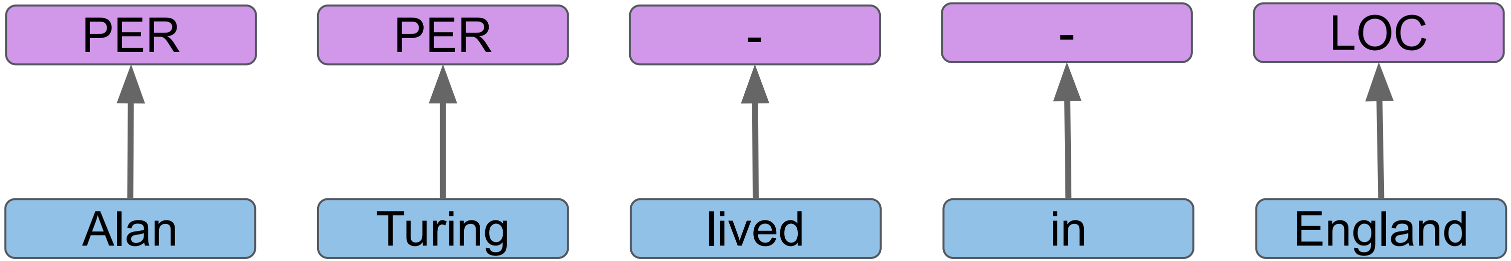
https://en.wikipedia.org/wiki/Natural_language_processing

(Some) NLP Tasks

- Classification (word-level)
  - **Named Entity Recognition**
  - Part of Speech Tagging
  - Extractive Question Answering
- Classification (sentence-level)
  - Sentiment Analysis
  - Spam Filters
- Classification (sentence pair-level)
  - Entailment
  - Sentence similarity
- Generative
  - Machine Translation
  - Abstractive Text Summarization
  - Abstractive Question Answering

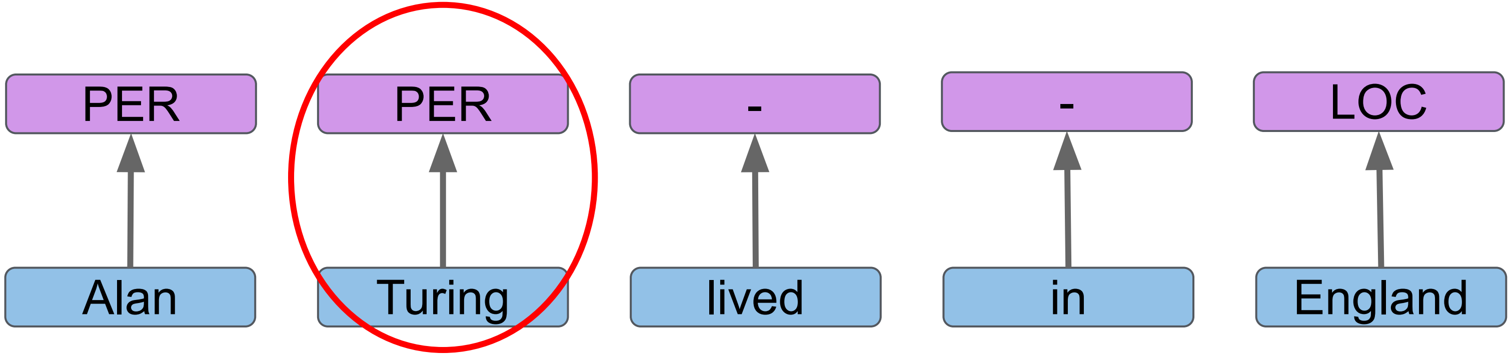
Named Entity Recognition

- Goal: assign a label to each **word**, describing its entity type.
  - E.g., let's assume that the list of labels is:
    - "-" (not a named entity),
    - "PER" (person),
    - "LOC" (location),
    - "TIME" (time).

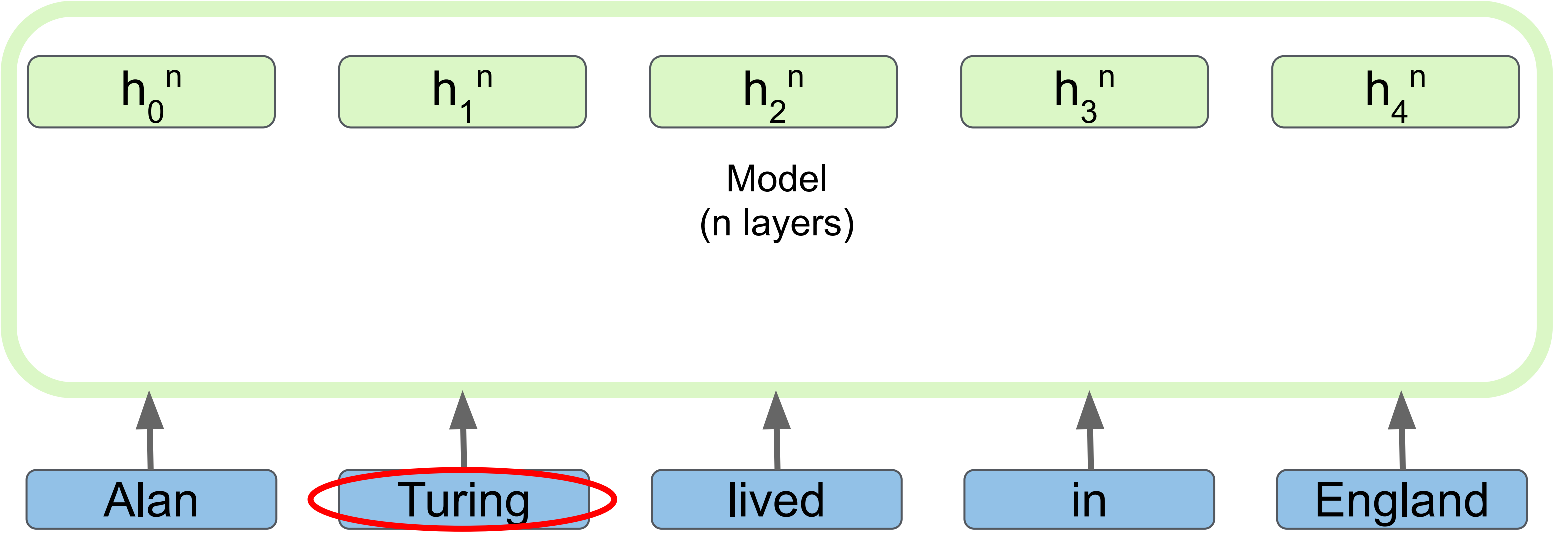


Named Entity Recognition

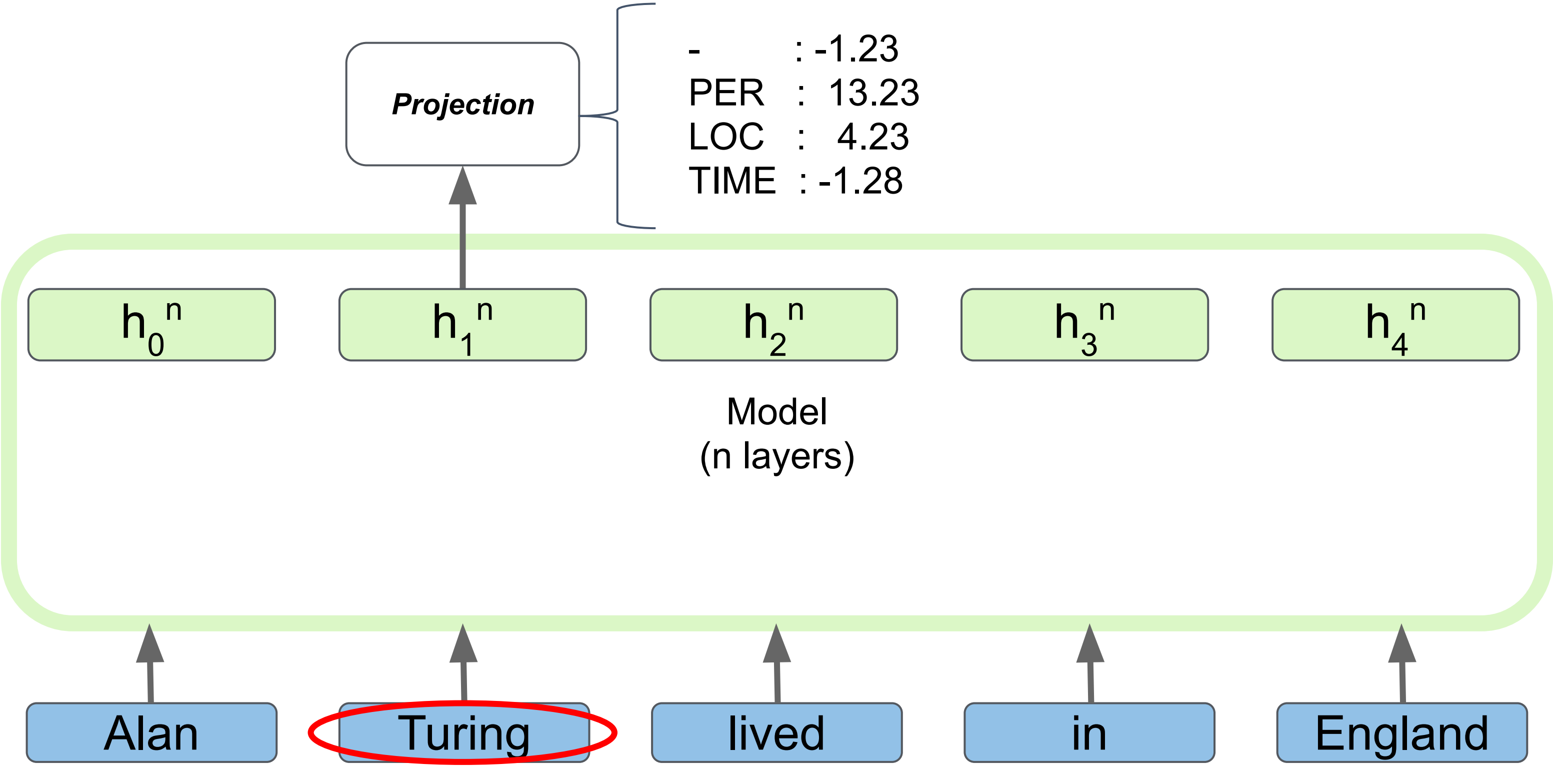
- Goal: assign a label to each **word**, describing its entity type.
    - E.g., let’s assume that the list of labels is: “-”, “PER”, “LOC”, “TIME”.
- Let’s focus our example on only one step (e.g., the word “Turing”).



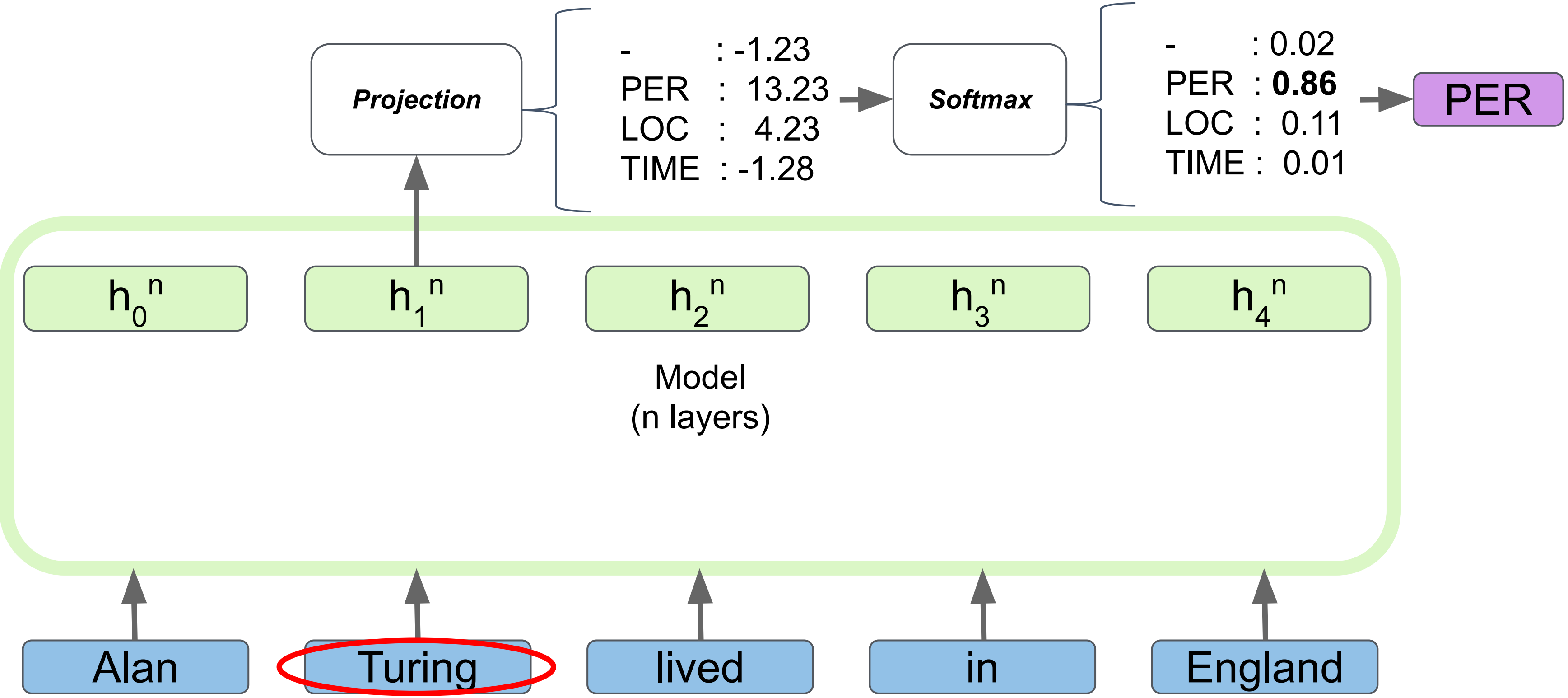
Named Entity Recognition



Named Entity Recognition



Named Entity Recognition

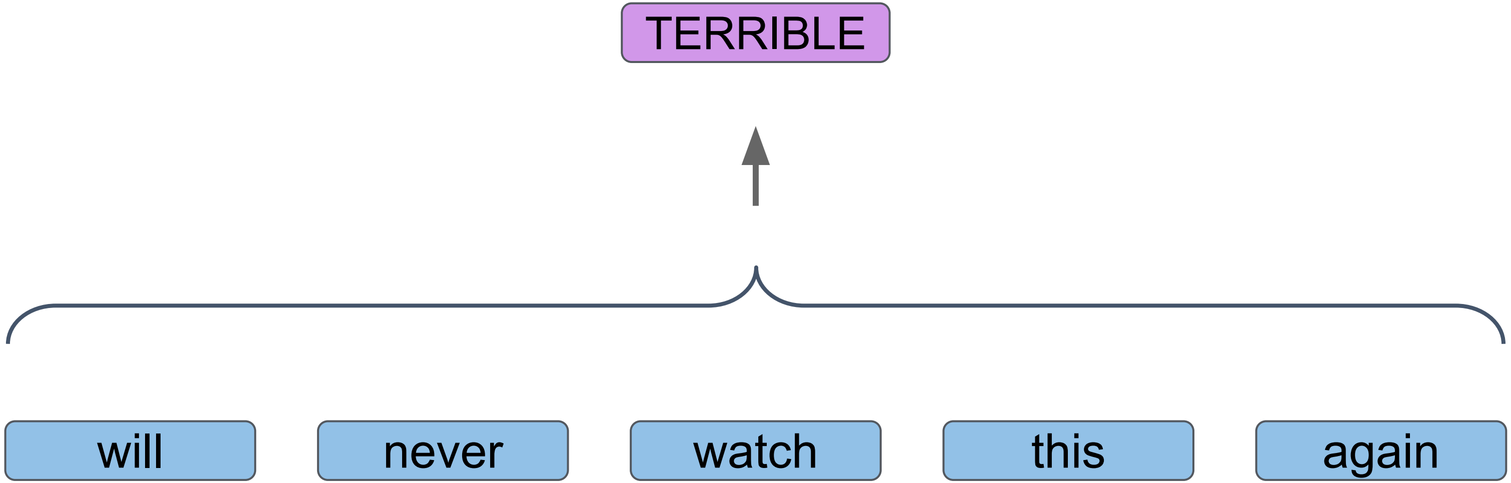


(Some) NLP Tasks

- Classification (word-level)
  - Named Entity Recognition
  - Part of Speech Tagging
  - Extractive Question Answering
- Classification (sentence-level)
  - **Sentiment Analysis**
  - Spam Filters
- Classification (sentence pair-level)
  - Entailment
  - Sentence similarity
- Generative
  - Machine Translation
  - Abstractive Text Summarization
  - Abstractive Question Answering

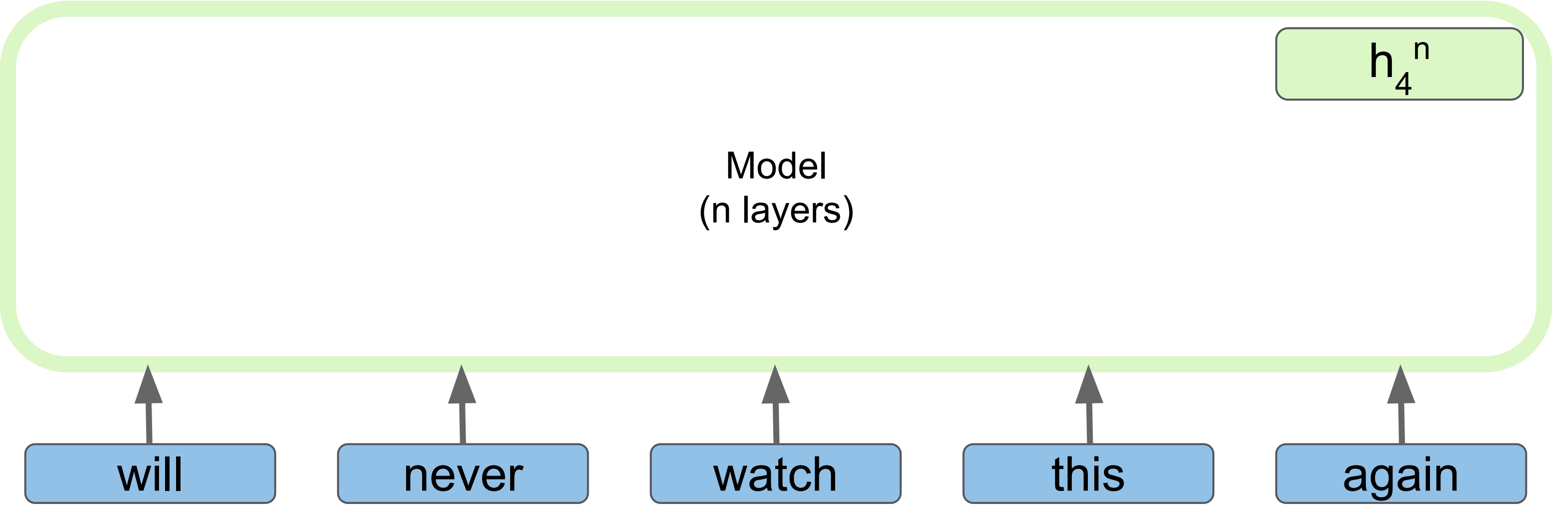
Sentiment Analysis

- Goal: assign one label to a **sentence** describing the sentiment that it conveys.



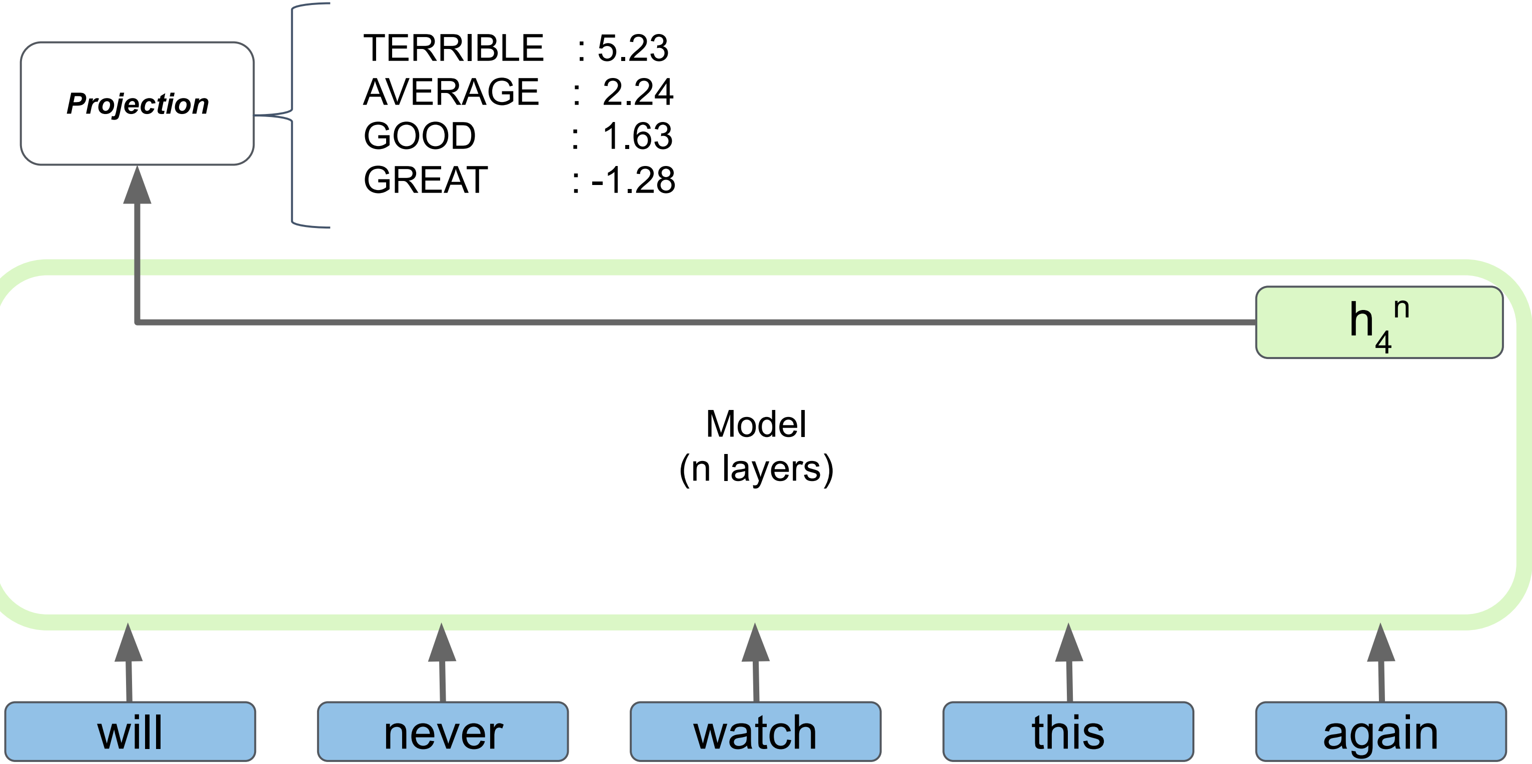
Sentiment Analysis

- Goal: assign one label to a **sentence** describing the sentiment that it conveys.
- Note that we only need the last model output (h_4^n).



Sentiment Analysis

- Goal: assign one label to a **sentence** describing the sentiment that it conveys.



Projection

TERRIBLE	: 5.23
AVERAGE	: 2.24
GOOD	: 1.63
GREAT	: -1.28

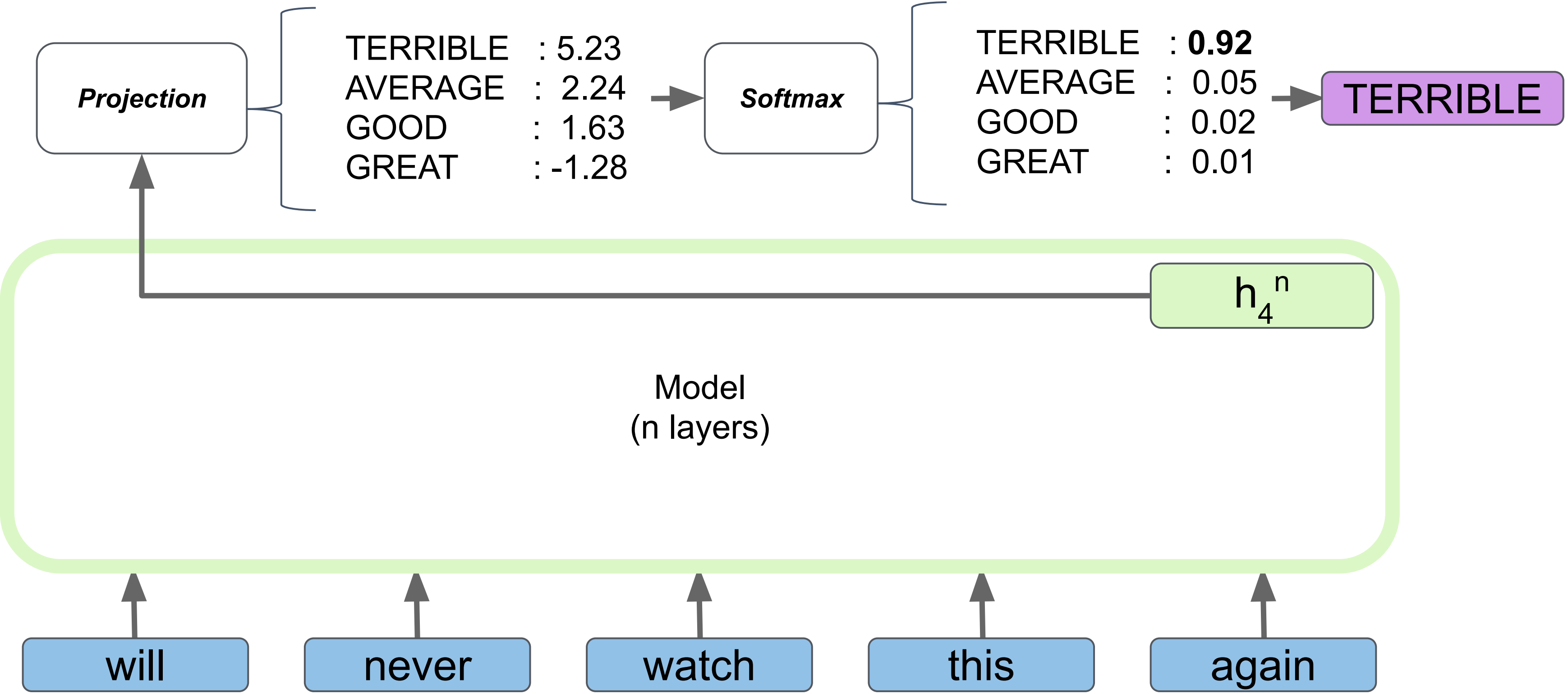
h_4^n

Model
(n layers)

will never watch this again

Sentiment Analysis

- Goal: assign one label to a **sentence** describing the sentiment that it conveys.



(Some) NLP Tasks

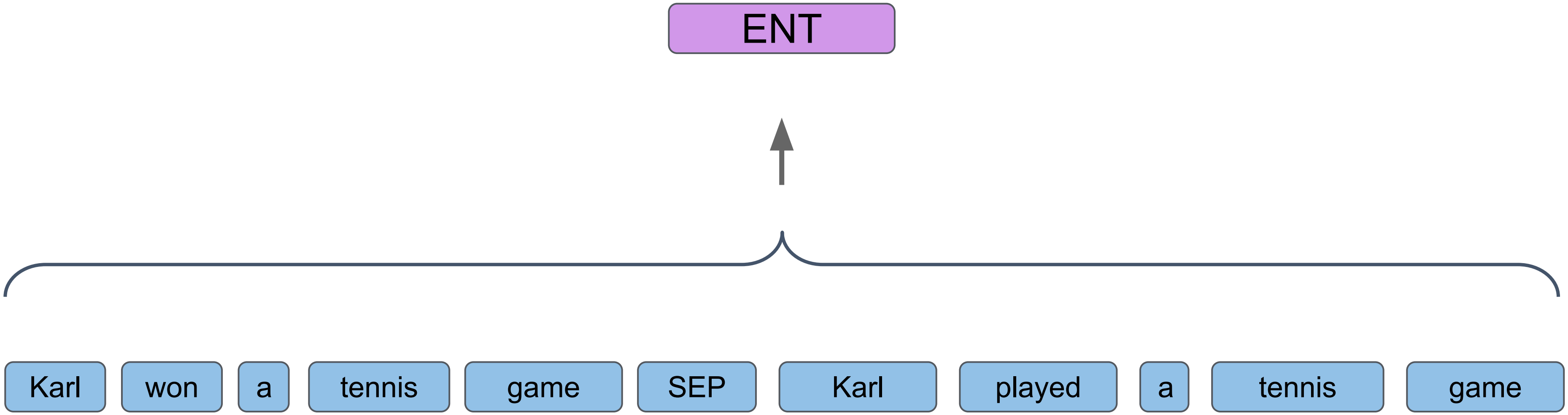
- Classification (word-level)
  - Named Entity Recognition
  - Part of Speech Tagging
  - Extractive Question Answering
- Classification (sentence-level)
  - Sentiment Analysis
  - Spam Filters
- Classification (sentence pair-level)
  - **Entailment**
  - Sentence similarity
- Generative
  - Machine Translation
  - Abstractive Text Summarization
  - Abstractive Question Answering

Entailment

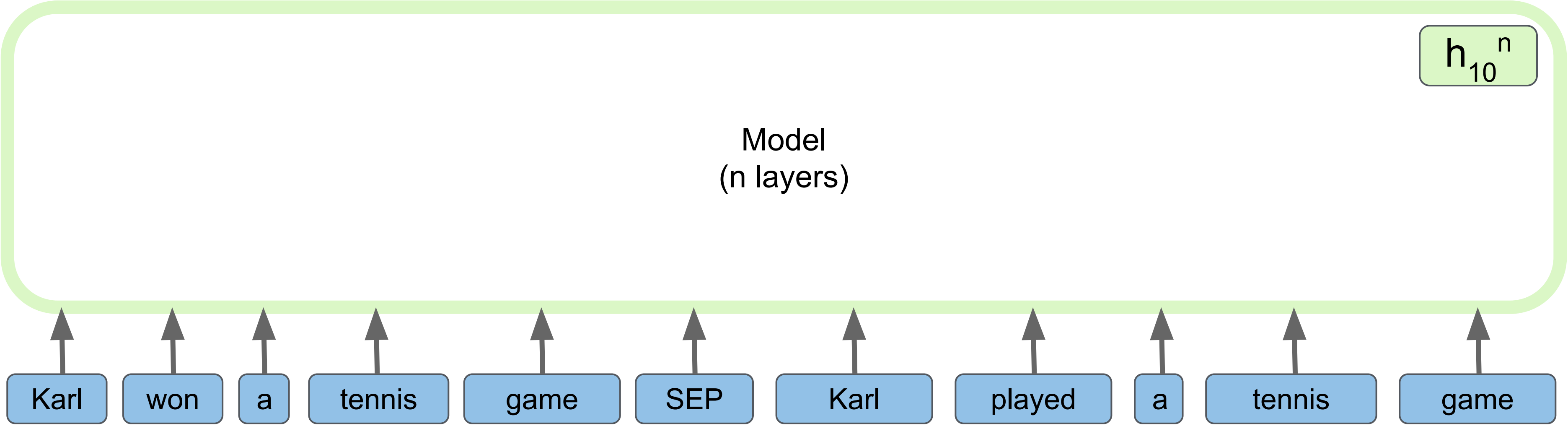
- Task: given two sentences, does the first one entail the second one? E.g.,
    - (Input) Sentence #1: "*Karl won a tennis game*"
    - (Input) Sentence #2: "*Karl played a tennis game*"
    - Target: "*entailment*"
- There are three possible labels:
    - "entailment"
    - "contradiction"
    - "neutral"

Entailment

SEP = separator indicating the end of the first sentence.

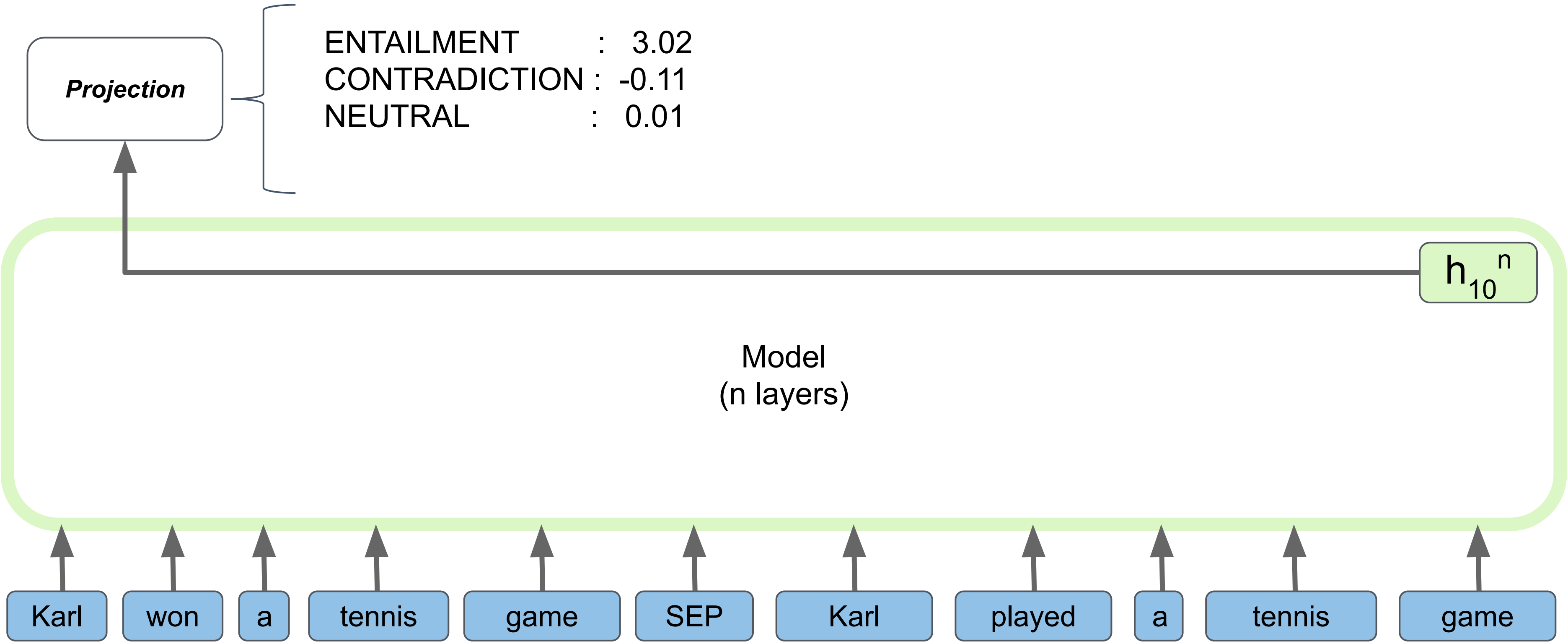


Entailment

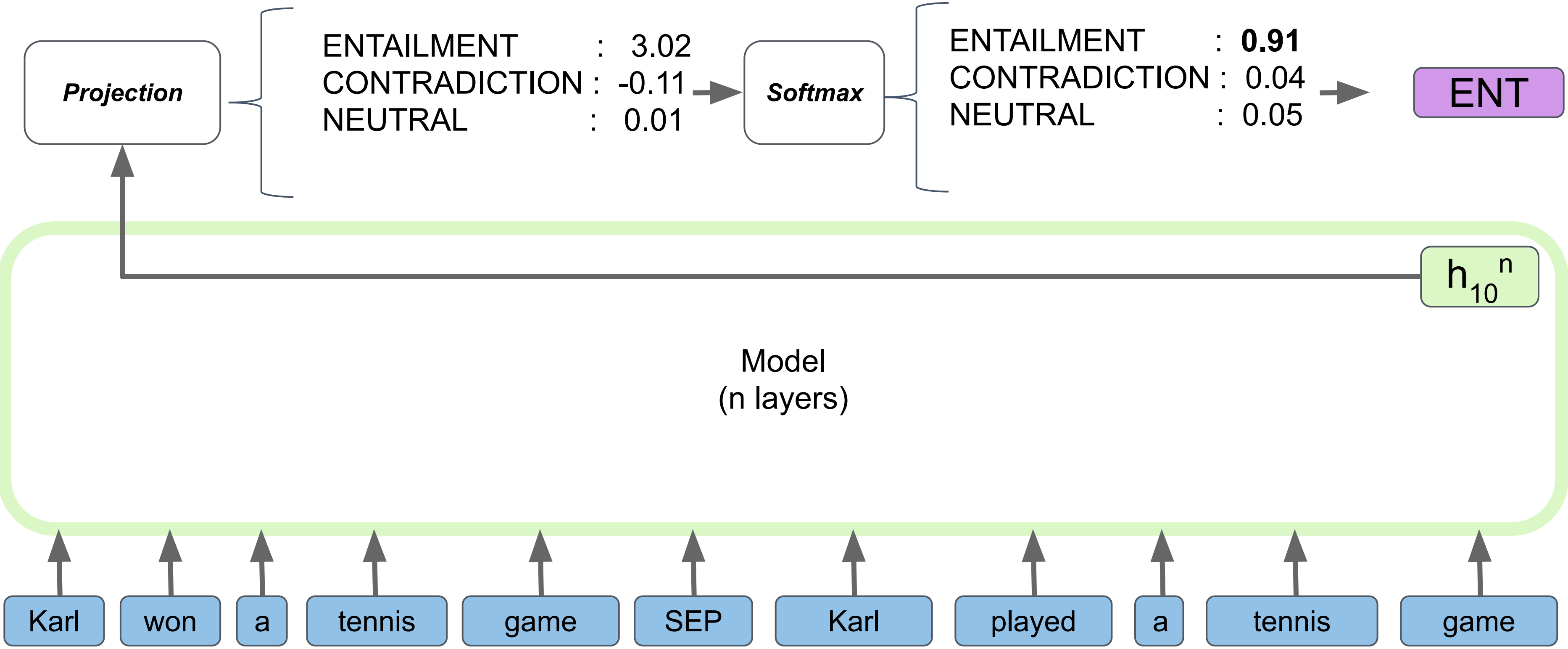


h_{10}^{n}
Model
(n layers)
Karl
won
a
tennis
game
SEP
Karl
played
a
tennis
game

Entailment


Projection
ENTAILMENT : 3.02
CONTRADICTION : -0.11
NEUTRAL : 0.01
h_{10}^{n}
Model
(n layers)
Karl
won
a
tennis
game
SEP
Karl
played
a
tennis
game

Entailment


Projection
ENTAILMENT : 3.02
CONTRADICTION : -0.11
NEUTRAL : 0.01
Softmax
ENTAILMENT : **0.91**
CONTRADICTION : 0.04
NEUTRAL : 0.05
ENT
h_{10}^{n}
Model
(n layers)
Karl
won
a
tennis
game
SEP
Karl
played
a
tennis
game

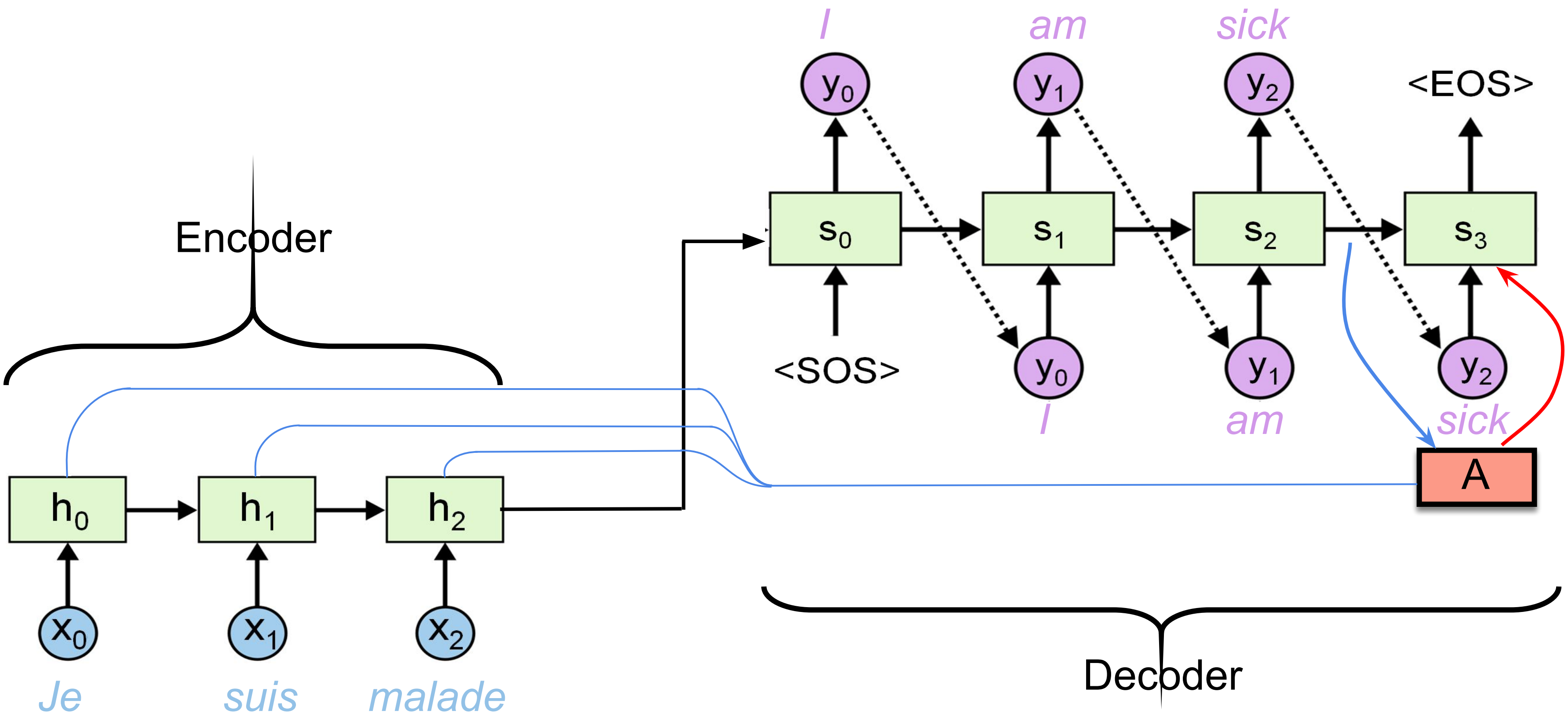
(Some) NLP Tasks

- Classification (word-level)
  - Named Entity Recognition
  - Part of Speech Tagging
  - Extractive Question Answering
- Classification (sentence-level)
  - Sentiment Analysis
  - Spam Filters
- Classification (sentence pair-level)
  - Entailment
  - Sentence similarity
- Generative
  - **Machine Translation**
  - Abstractive Text Summarization
  - Abstractive Question Answering

Machine Translation

- Task: given a sentence, translate it into another language. E.g.,
  - Input: “*Je suis malade*”
  - Target: “*I am sick*”
- Machine translation requires a sequence-to-sequence (encoder plus decoder) model.
  - The encoder parses the input.
  - The decoder produces the output (using an autoregressive approach).
  - Attention (between encoder and decoder) greatly improves results.
  - (We saw all these components in the previous presentation.)

Machine Translation



Plan

- Natural Language Processing
- **Words and Semantics**
- Classical Approaches
- Word Embeddings
- Contextualized Word Embeddings
- BERT
- Summary

Words and Semantics

- In all NLP tasks, we need to “access” the word/sentence semantics in order to solve a given task.
- Finding a link between words and semantics is not always trivial.

Words and Semantics - Example

- Example: question answering.

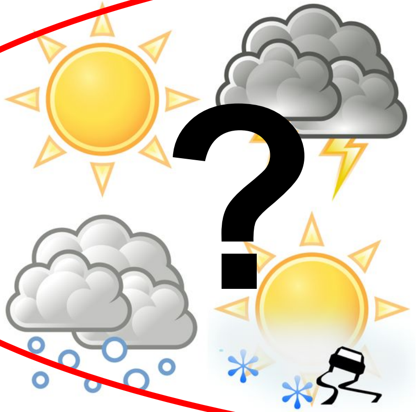
How is the weather now?

It's raining cats and dogs.

https://commons.wikimedia.org/wiki/File:Weather_symbols_p.png
https://unsplash.com/photos/F-t5EpfQNpk

Words and Semantics - Example

- Note how we are interested in the semantics...



How is the weather now?

It’s raining cats and dogs.

https://commons.wikimedia.org/wiki/File:Weather_symbols_p.png
https://unsplash.com/photos/F-t5EpfQNpk

Words and Semantics - Example

- Note how we are interested in the semantics…
- ...but we only have access to the words.

How is the weather now?

It’s raining cats and dogs.

https://commons.wikimedia.org/wiki/File:Weather_symbols_p.png
https://unsplash.com/photos/F-t5EpfQNpk

Linking Words and Semantics

- We need some way to link words and semantics:
  - Distributional Hypothesis
  - Principle of Compositionality

Distributional Hypothesis

- “Words that occur in the same contexts tend to have similar meanings (Harris, 1954).”
- “You shall know a word by the company it keeps” Firth, J. R. 1957:11

- The distributional hypothesis suggests that **the more semantically similar** two words are, **the more distributionally similar** they will in turn be, and thus the more that they will tend to occur in similar linguistic contexts.

https://en.wikipedia.org/wiki/Distributional_semantics

Distributional Hypothesis

- *“Through their intelligence, humans possess the cognitive abilities to learn, form concepts [...]”*
- *“Intelligence is what makes humans the most successful [...]”*
- *“Human intelligence is essential to better understand [...]”*

- Note how the word “human” is often in the context of the word “intelligence”.

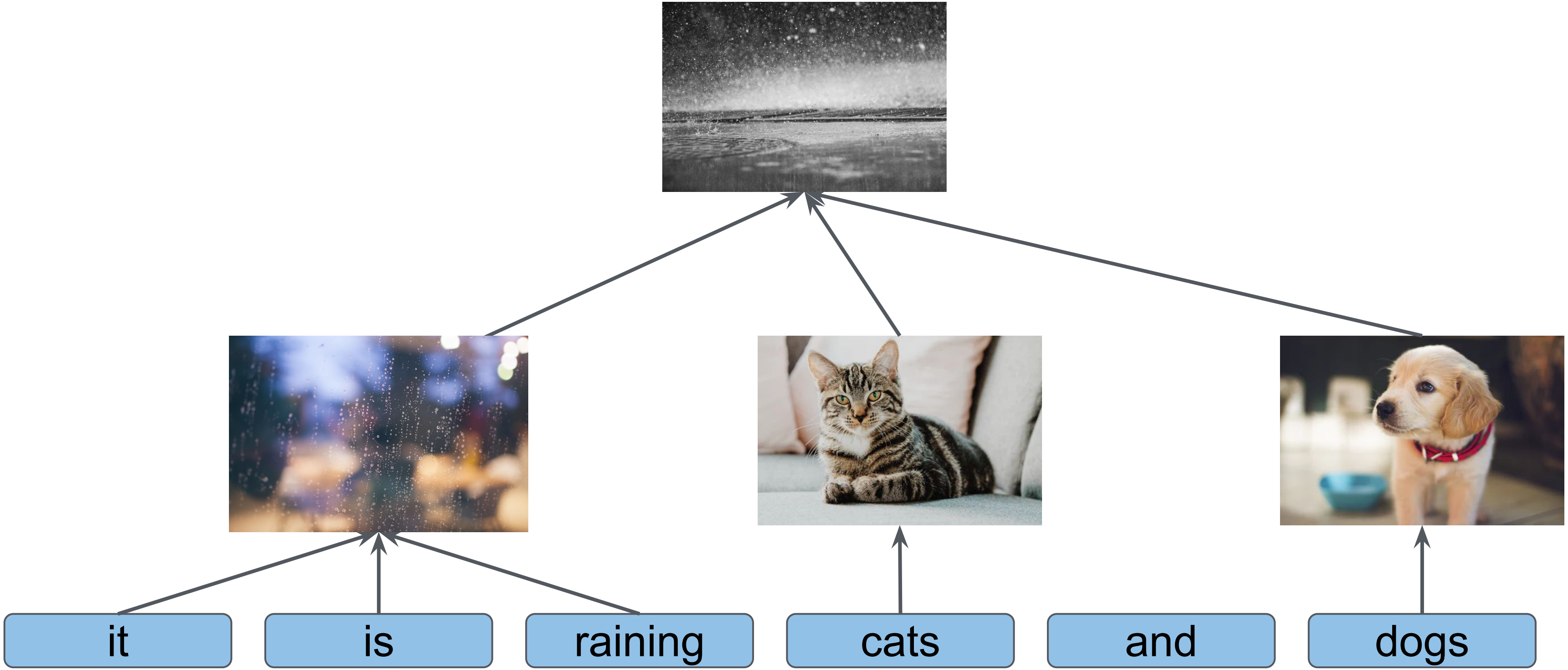
https://en.wikipedia.org/wiki/Human_intelligence

Principle of compositionality

- "In mathematics, semantics, and philosophy of language, the principle of compositionality is the principle that the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them."

https://en.wikipedia.org/wiki/Principle_of_compositionality

Principle of compositionality



it is raining cats and dogs

https://unsplash.com/photos/ngqyo2AYYnE, https://unsplash.com/photos/IbPxGLgJiMI, https://unsplash.com/photos/VR0s3Yqm2RA, https://unsplash.com/photos/F-t5EpfQNpk

Linking Words and Semantics

- The distributional hypothesis is a promising way to generate semantics for a word by looking at the context where the word appears.
- The principle of compositionality allows us to tackle the NLP tasks in a hierarchical way.
- Both can help us in creating algorithms to solve NLP tasks.

- Before doing so though, we can start by looking at older/classical approaches to better understand the evolution of some NLP algorithms.

Plan

- Natural Language Processing
- Words and Semantics
- **Classical Approaches**
- Word Embeddings
- Contextualized Word Embeddings
- BERT
- Summary

Tokens

- A token is a unit in a text sequence.

- It can be composed of:
  - characters, e.g., “a”, “b”, ...
  - subwords, e.g., “er”, “est”, ...
  - words, e.g., “cat”, “house”, ...

Tokens

- A token is a unit in a text sequence.

- It can be composed of:
  - characters, e.g., “a”, “b”, ...
  - subwords, e.g., “er”, “est”, ...
  - words, e.g., “cat”, “house”, ...

- All three representations have advantages and disadvantages.
- We will now compare them with respect to:
  - vocabulary size,
  - sentence length,
  - handling of out-of-vocabulary tokens.

Tokens - Vocabulary Size

- The longer the token string, the bigger the vocabulary.
- words > subwords > characters
    - E.g.,

words:	~80k
subwords:	~20k
characters:	< 100

- A small vocabulary is usually better (both computationally and result wise).

Tokens - Sentence Length

- The smaller the token unit, the longer the representation of a sentence.
- characters > subwords > words
  - E.g.,

“hi there”	(2 words)
“hi _ the re”	(4 subwords)
“h i _ t h e r e”	(8 characters)

- A short representation is usually better (both computationally and result wise).

Tokens - Out Of Vocabulary

- An out-of-vocabulary (OOV) token is a token that does not appear in training.

- In general, there is a big chance that the training data will not contain all the possible words for a given task.
  - The amount of OOV tokens can be quite large when using word-based vocabularies.

- On the other hand, the chance that a subword or a character does not appear in training is very small.
  - OOV tokens are very few (or absent) when using subwords or characters.

Tokens - Summary

- Every choice (word / subword / character) has its own advantages and disadvantages.

- Subwords are usually a good compromise that avoids the two extreme cases of having a too big vocabulary and having too long sentences.
- They are also less affected by the OOV problem.

- In the rest of this presentation, for the sake of simplicity, we will assume that a token is a word.

Token Representation

- In a **one hot representation**, each token is associated with a vector whose elements are all equal to 0, except one, which has a value equal to 1.
  - The vector size is equal to the vocabulary size.
  - The vector is formed of bits (which can take values 0 / 1), with each bit corresponding to a specific word.

- Example #1 - with a vocabulary of 3 words (‘cat’, ‘dog’, ‘house’):

  ‘cat’ = [0, 0, 1]
  ‘dog’ = [0, 1, 0]
  ‘house’ = [1, 0, 0]

Token Representation

- Example #2 - with a vocabulary of 4 words (‘cat’, ‘dog’, ‘house’, ‘pc’):

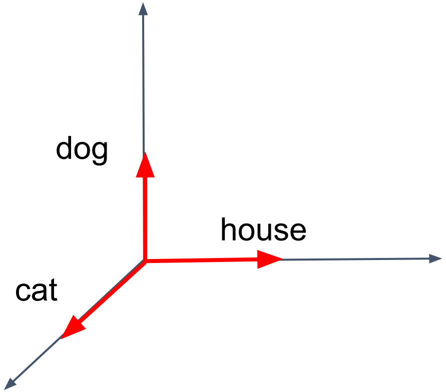
‘cat’	=	[0, 0, 0, 1]
‘dog’	=	[0, 0, 1, 0]
‘house’	=	[0, 1, 0, 0]
‘pc’	=	[1, 0, 0, 0]

- Note how vector size = vocabulary size
  (this can be a problem if the vocabulary size is big)

Token Representation

- Words represented as one hot vectors are all at the same distance of each other.
- This is not desirable.
- For example, we would prefer to have ‘dog’ and ‘cat’ closer to each other than they are to ‘house’ (which is not the case here)

‘cat’	=	[0, 0, 1]
‘dog’	=	[0, 1, 0]
‘house’	=	[1, 0, 0]



Sentence Representation

- A **bag of word representation** corresponds to the sum of one hot vectors.

- Example: given the following one hot vectors
  ‘cat’ = [0, 0, 1]
  ‘dog’ = [0, 1, 0]
  ‘house’ = [1, 0, 0]

  the bag of word representation of the sequence ‘cat dog house’ is:
  ‘cat dog house’ = [1, 1, 1]

Sentence Representation

- A **bag of word representation** corresponds to the sum of one hot vectors.

- Example: given the following one hot vectors
  ‘cat’ = [0, 0, 1]
  ‘dog’ = [0, 1, 0]
  ‘house’ = [1, 0, 0]

  the bag of word representation of the sequence ‘cat dog house’ is:
  ‘cat dog house’ = [1, 1, 1]

- The order of the words is lost!
- E.g., “drink water not poison” = “drink poison not water”

Plan

- Natural Language Processing
- Words and Semantics
- Classical Approaches
- **Word Embeddings**
- Contextualized Word Embeddings
- BERT
- Summary

Beyond One Hot Vectors

- One hot vectors are **local** representations.
- They represent a 1:1 mapping between a word (e.g., ‘cat’, ‘dog’, ‘house’) and a position in a vector (i.e., the position with the value of 1).

Hinton, McClelland, Rumelhart, “Parallel distributed processing: explorations in the microstructure of cognition, vol. 1”

Beyond One Hot Vectors

- This representation has two main disadvantages:
  - The vector dimension is proportional to the vocabulary size.
  - It does not capture relationships between words.

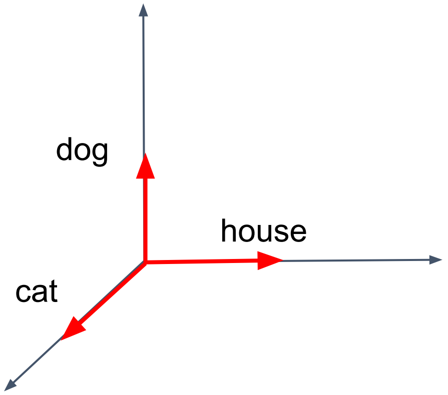
- E.g., with a vocabulary containing 80,000 words:

$$\text{‘dog’} = \overbrace{[0, 0, 0, 0, 0, 0, 0, 0, \ldots\ 1, \ldots\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}^{80{,}000}$$

Beyond One Hot Vectors

- This representation has two main disadvantages:
  - The vector dimension is proportional to the vocabulary size.
  - It does not capture relationships between words.
    - All words are at the same distance.

‘cat’	=	[0, 0, 1]
‘dog’	=	[0, 1, 0]
‘house’	=	[1, 0, 0]



Distributed Representation

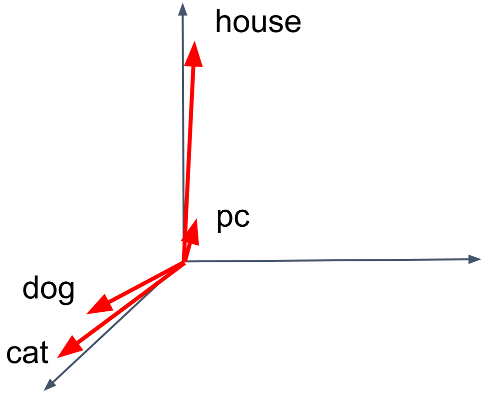
- A different approach is to distribute the representation of a word across all available dimensions in a continuous space in a way that leads to “similar” words being close to each other.
- This is called a **distributed** representation.

Hinton, McClelland, Rumelhart, “Parallel distributed processing: explorations in the microstructure of cognition, vol. 1”

Distributed Representation

- Let’s assume that the space dimensionality is 3 and that we have the following distributed representations:

‘cat’	=	[1.1, 0.46, 45]
‘dog’	=	[1.1, 1.10, 30]
‘house’	=	[0.2, 4.51, 0]
‘pc’	=	[0.1, 0.3 , 0]

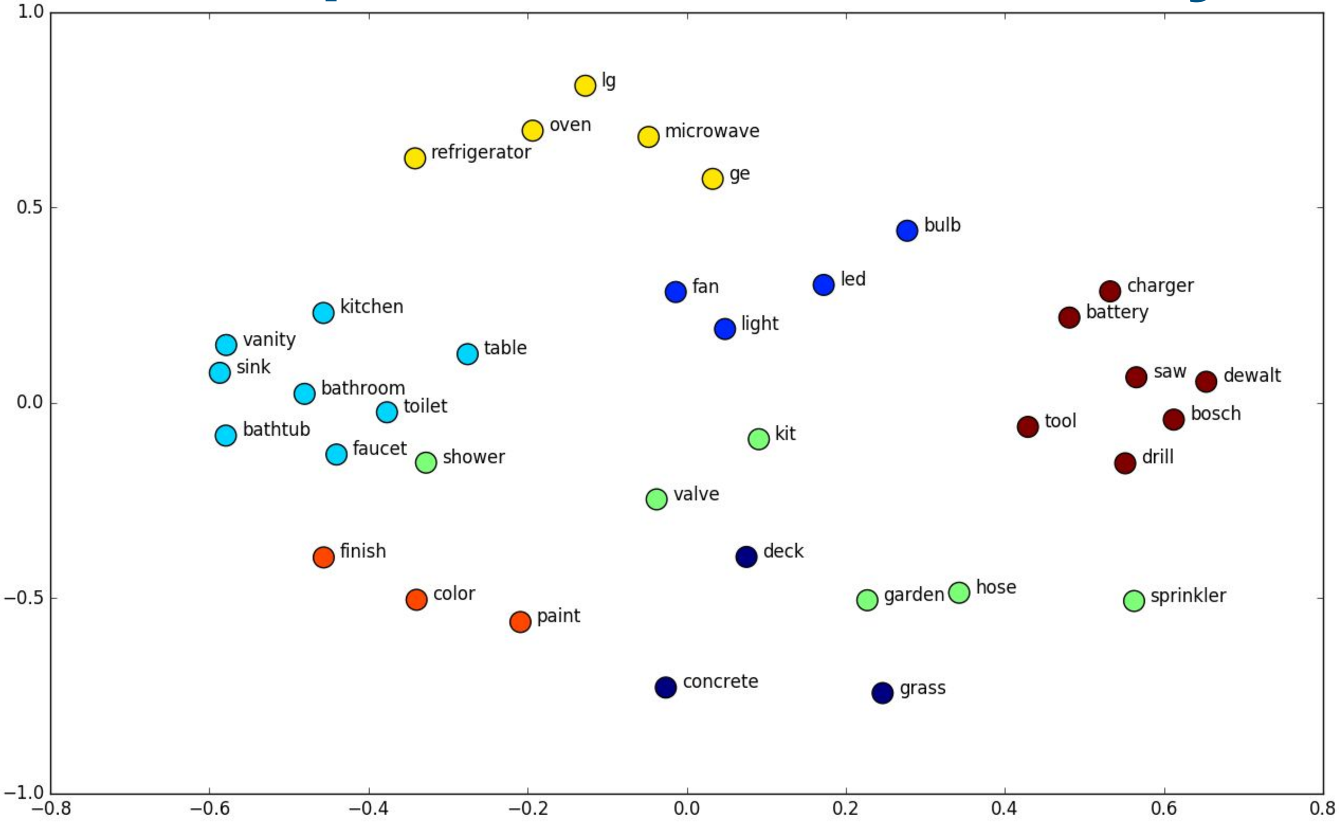


- Note how the vector size is **not** equal to the vocabulary size.
  - The vector size will in general be much smaller than the vocabulary size.
- Also note how ‘dog’ and ‘cat’ are more “similar” than ‘dog’ and ‘house’.

Distributed Representation

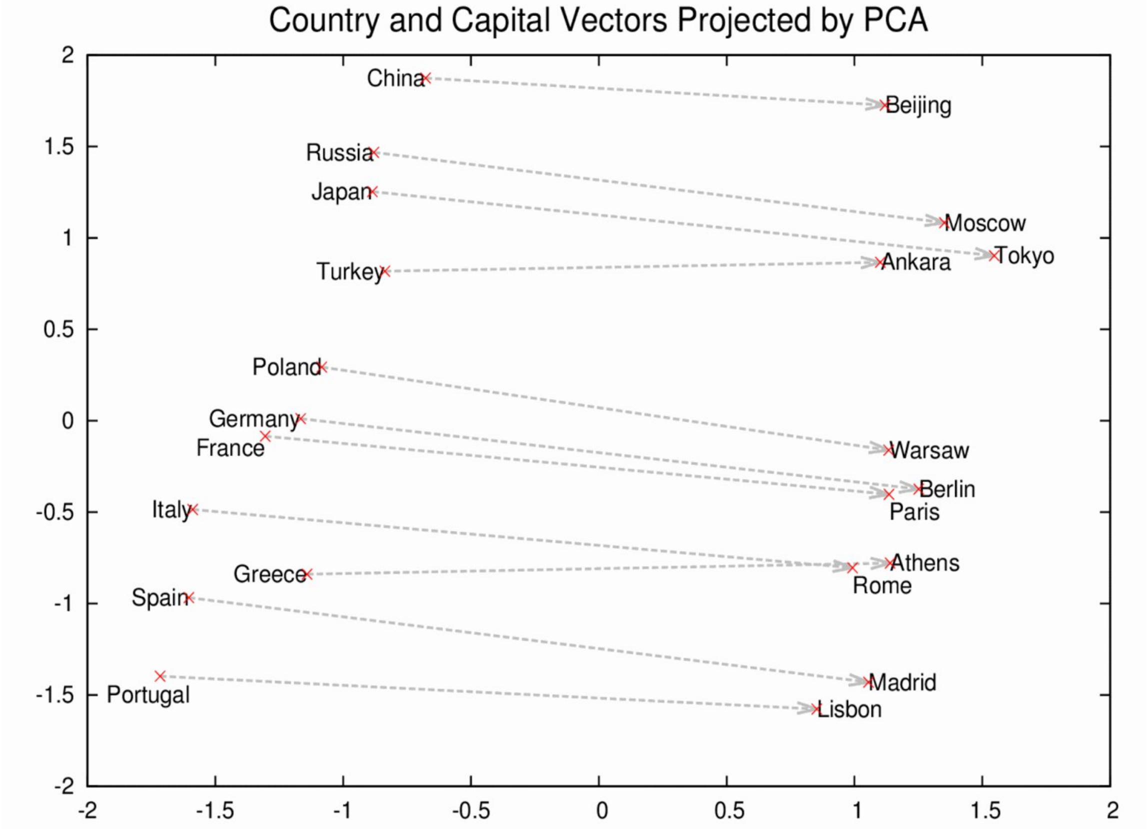
- Distributed representations can expose useful properties:
  - Semantically similar words can be close to each other.
  - Relationships between words can be captured.

Distributed Representation - Similarity



https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Distributed Representation - Relationships



Mikolov et al., “Distributed Representations of Words and Phrases and their Compositionality”

Word Embeddings

- Distributed representations for words are called **word embeddings**.
- There are several ways to generate word embeddings that expose the desired properties that we mentioned, including:
  - Word2Vec
  - FastText

Word2Vec

- Word2Vec is an efficient way to compute word embeddings.
- It is inspired from the distributional hypothesis.
- It is based on a very simple linear model.
- There are two versions:
    - Continuous bag-of-words.
    - Continuous Skip-Gram.

Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec - Continuous bag-of-words

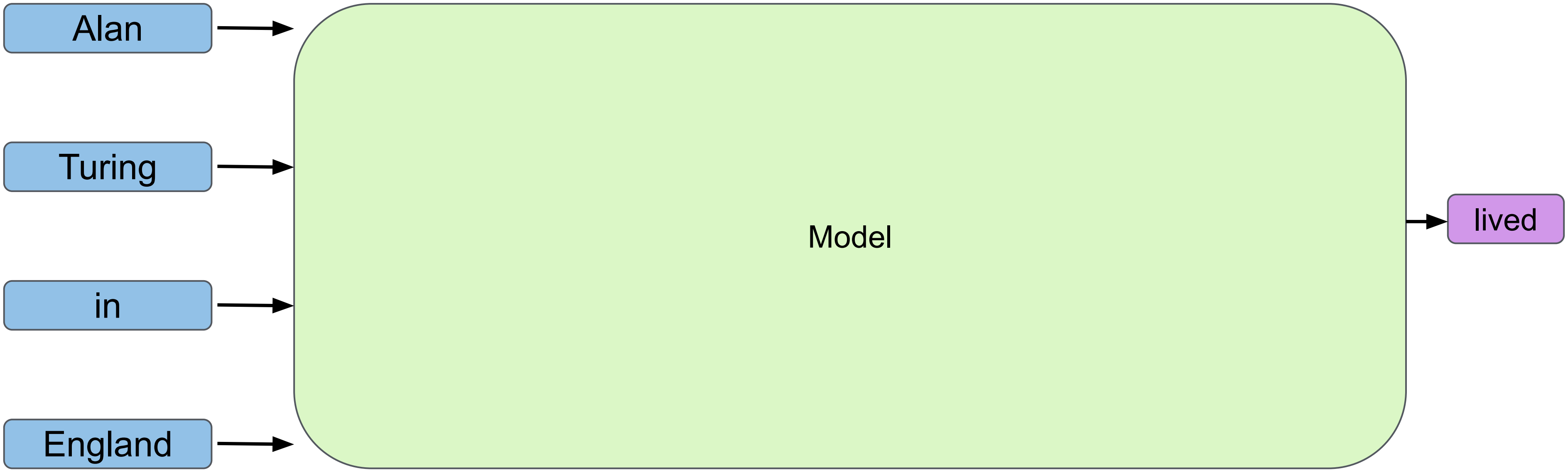
- The task is simple:
  - Mask a word in a sequence of text.
  - Train a model to predict this word given the words that surround it.

- For example, for “Alan Turing lived in England”:
  - Input: “Alan Turing in England”
  - Target: “lived”

Mikolov et al. "Efficient estimation of word representations in vector space."

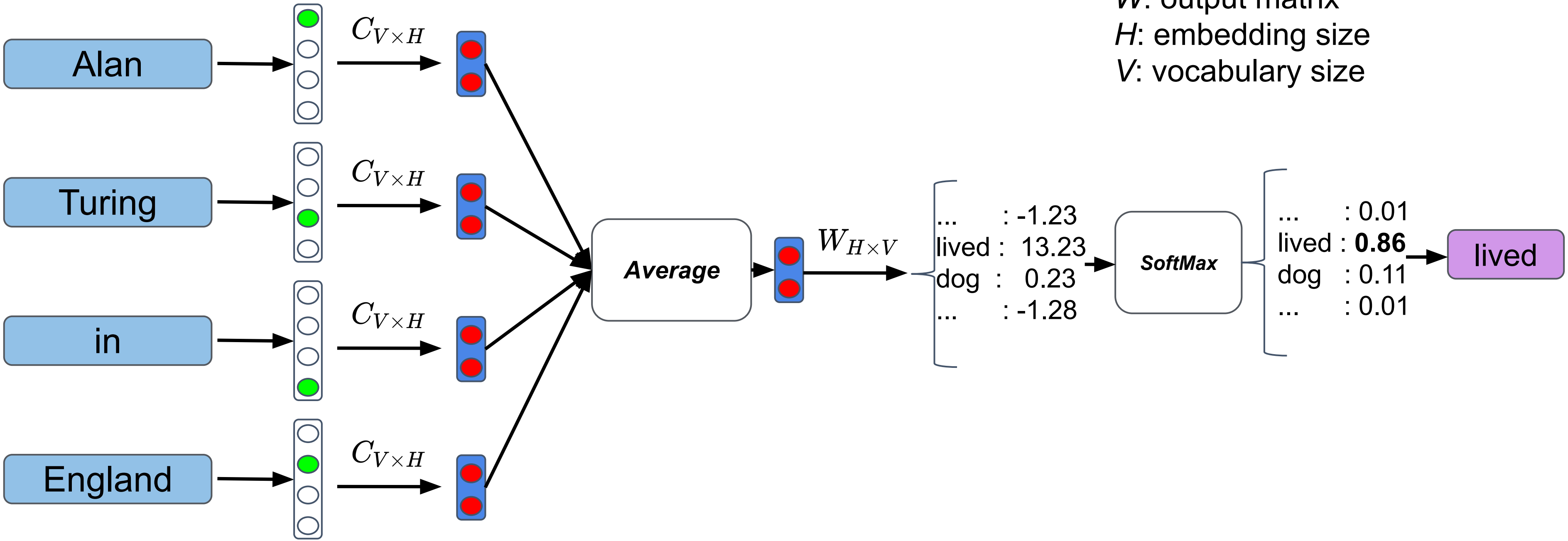
Word2Vec - Continuous bag-of-words

- Task: predict a word given some context window.



Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec - Continuous bag-of-words



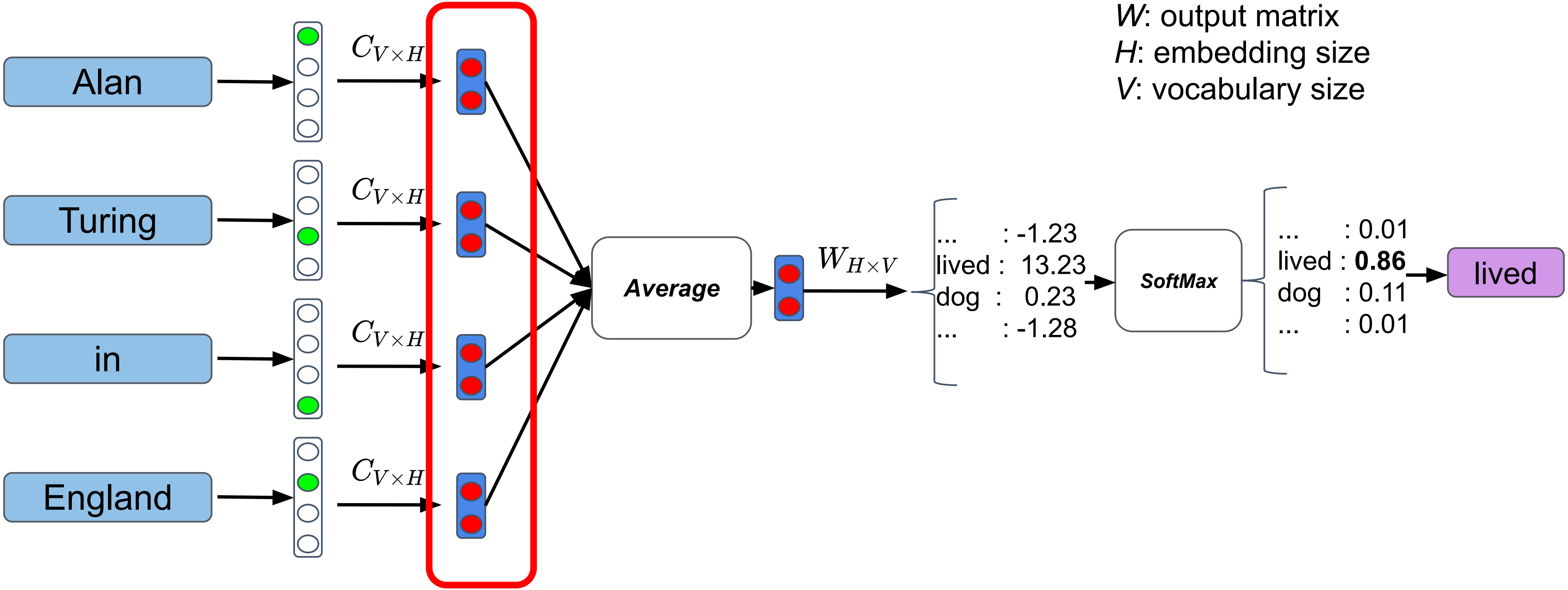
C: input matrix
W: output matrix
H: embedding size
V: vocabulary size

Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec - Continuous bag-of-words

- The embeddings is what we really care about.

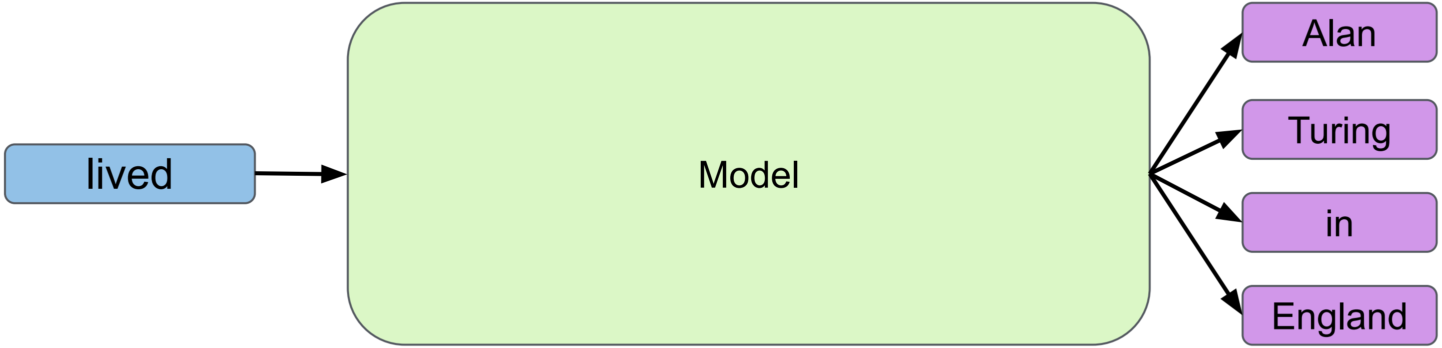
C: input matrix
W: output matrix
H: embedding size
V: vocabulary size



Mikolov et al. "Efficient estimation of word representations in vector space."

Word2Vec - Continuous Skip-Gram

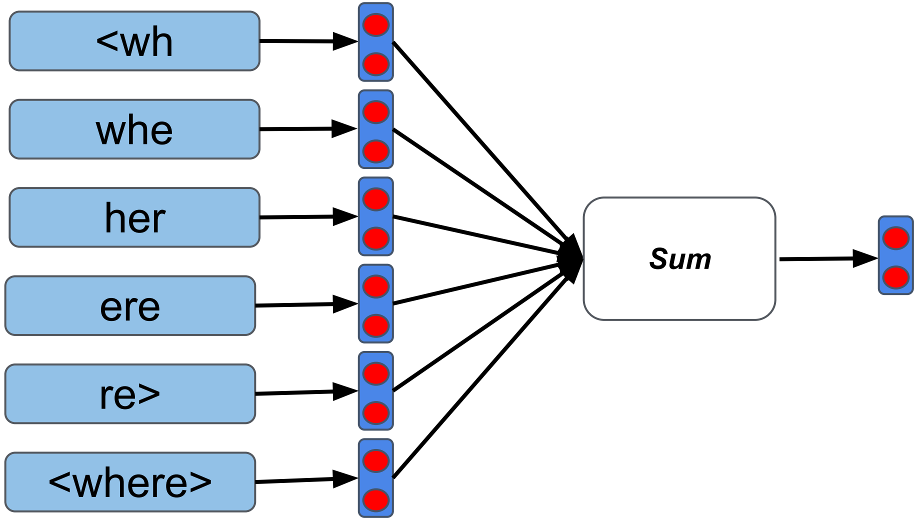
- In the Continuous Skip-Gram version, the Word2Vec algorithm has a different goal:
    - It predicts the context given the central word.
- It works better than continuous bag-of-words in the presence of smaller amounts of data.



Mikolov et al. "Efficient estimation of word representations in vector space."

FastText

- Extension of the Continuous Skip-Gram model to:
    - better handle out-of-vocabulary words,
    - consider the morphology of the words.
- The final embedding for one word is the sum of the original word embedding and the embeddings of its n-grams.



FastText representation of the word "where" if we only use 3-grams

Bojanowski et al. "Enriching Word Vectors with Subword Information."

Pre-Training

- Prior to training a model on a task of interest, the embeddings can be learned in a pre-training phase.

- Pre-training can be greatly beneficial given that algorithms like Word2Vec and FastText can learn embeddings in an unsupervised way, thus making it possible to leverage huge amounts of data.

Pre-Training

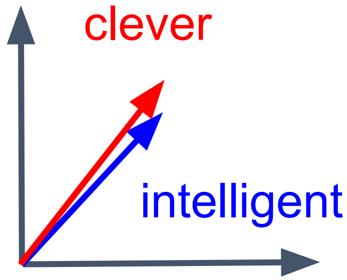
- Once pre-training is done, the embeddings can then used for the real task of interest (i.e., the training phase).
  - This task is usually supervised and cannot therefore leverage large amounts of data.
- This idea greatly helps to deal with scarcity of data in the training phase.
- The systems we will see next strongly build on this approach.

Plan

- Natural Language Processing
- Words and Semantics
- Classical Approaches
- Word Embeddings
- **Contextualized Word Embeddings**
- BERT
- Summary

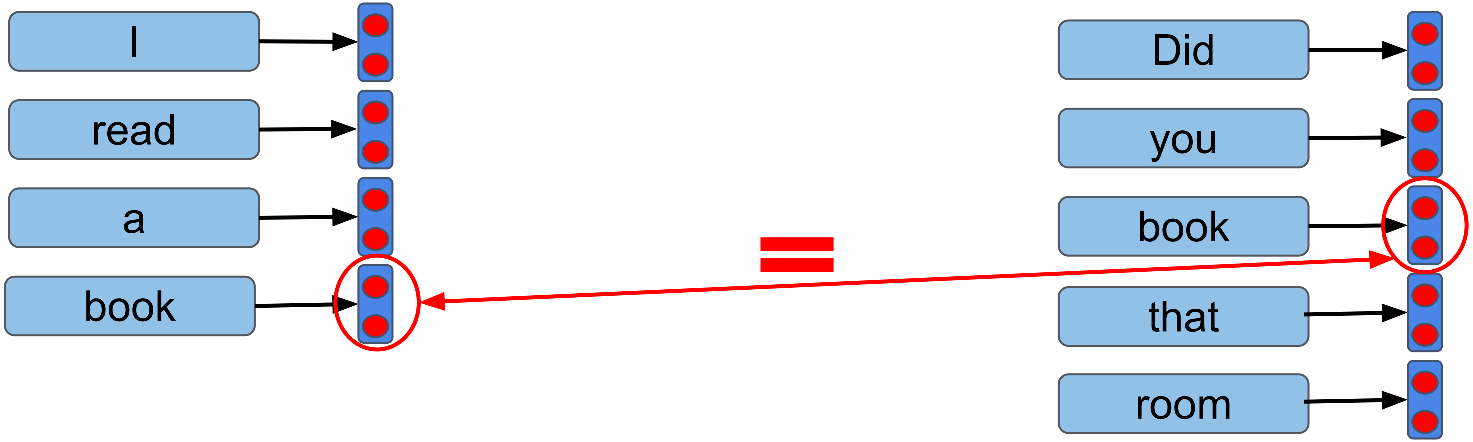
Word Embeddings and Synonymy

- Word embeddings are very useful representations that often lead to improved results.
- This is in part due to the fact that they can capture synonymy properties.
  - Synonyms in general have very close embedding vectors.



Word Embeddings and Polysemy

- Word embeddings do **not** help with polysemy though.
- A word such as "*book*" has the **same** embedding **regardless of the context**.



Word Embeddings and Polysemy

- Word embeddings reply to the question:

  what is the embedding for “book”?

- In order to consider the context, the question should become:

  what is the embedding for “book” in the sentence “I read a book”?

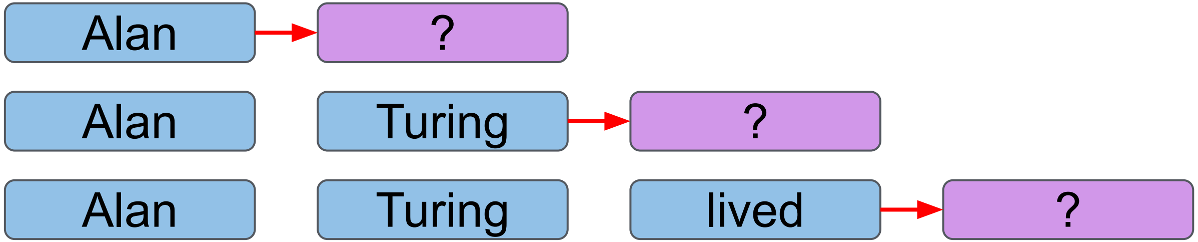
ELMo

- ELMo (Embeddings from Language Models) proposes a solution to two problems:
  - Generate **contextualized** word embeddings...
  - … in an unsupervised pre-training phase.
- To pre-train, they select the Language Modeling (LM) task.

Peters et al. "Deep contextualized word representations."

Language Modeling

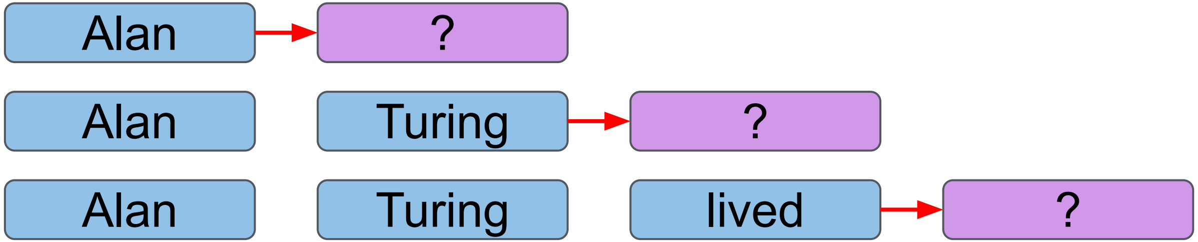
- Given n words (from a sentence), predict the next word (n+1).



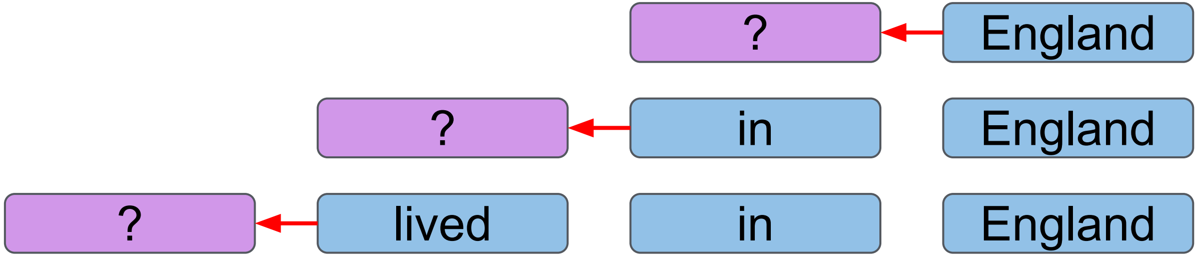
- Language Modeling is an unsupervised task.

Language Modeling

- Given n words (from a sentence), predict the next word (n+1).

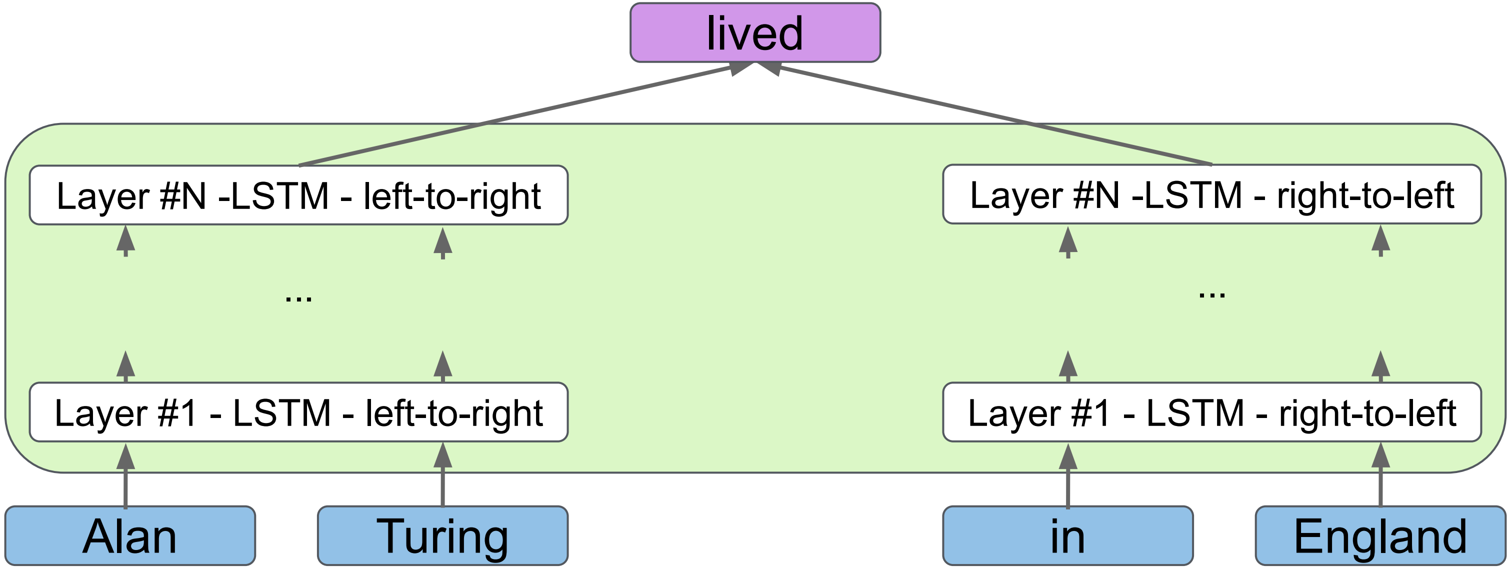


- Language Modeling is an unsupervised task.
- Note it can also work ‘right-to-left’.



ELMo

- ELMo model is a N-layer bidirectional LSTM.
- The contextualized embedding at a time step corresponds to a combination of the hidden states of all the various layers.

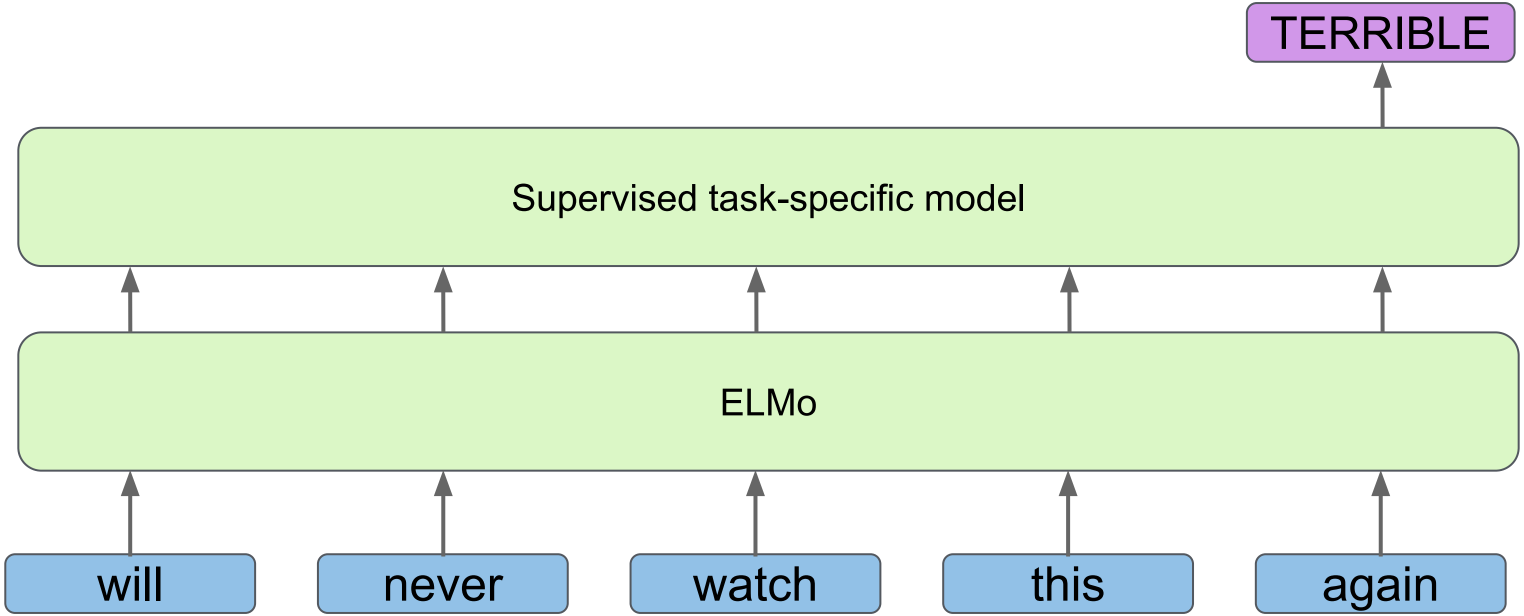


lived
Layer #N -LSTM - left-to-right
Layer #N -LSTM - right-to-left
...
...
Layer #1 - LSTM - left-to-right
Layer #1 - LSTM - right-to-left
Alan
Turing
in
England

Peters et al. "Deep contextualized word representations."

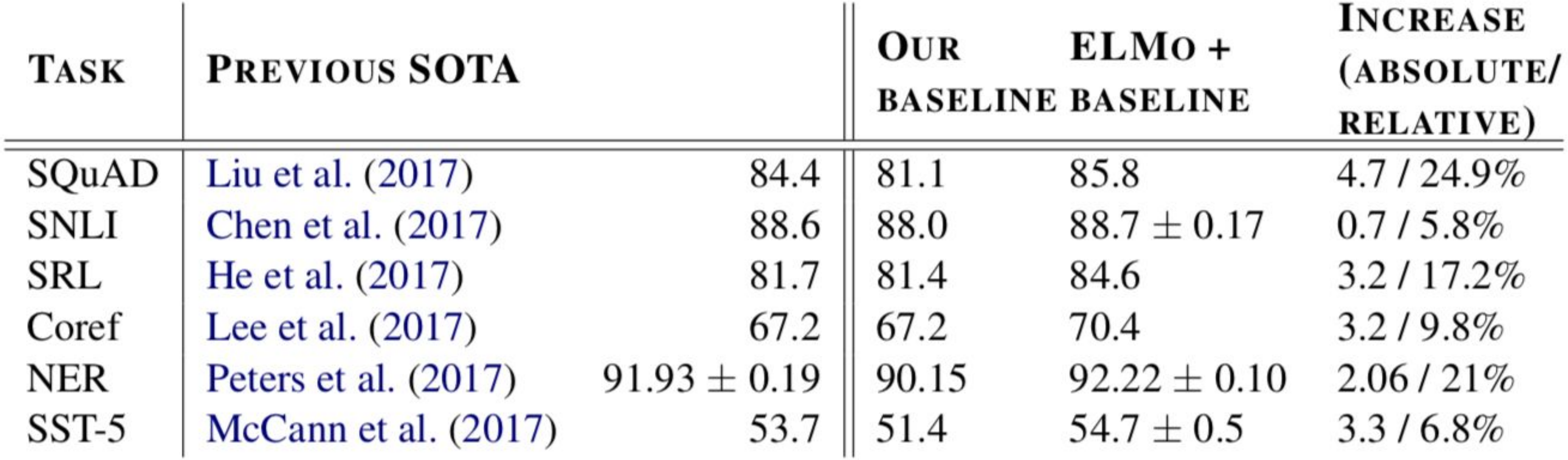
ELMo

- After pre-training, ELMo can be used with any other model (trained on supervised task).



Peters et al. "Deep contextualized word representations."

ELMo - Results



Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

SQuAD: (F1) extractive question answering.

SNLI: (accuracy) textual entailment.

SRL: (F1) semantic role labeling (word-based classification).

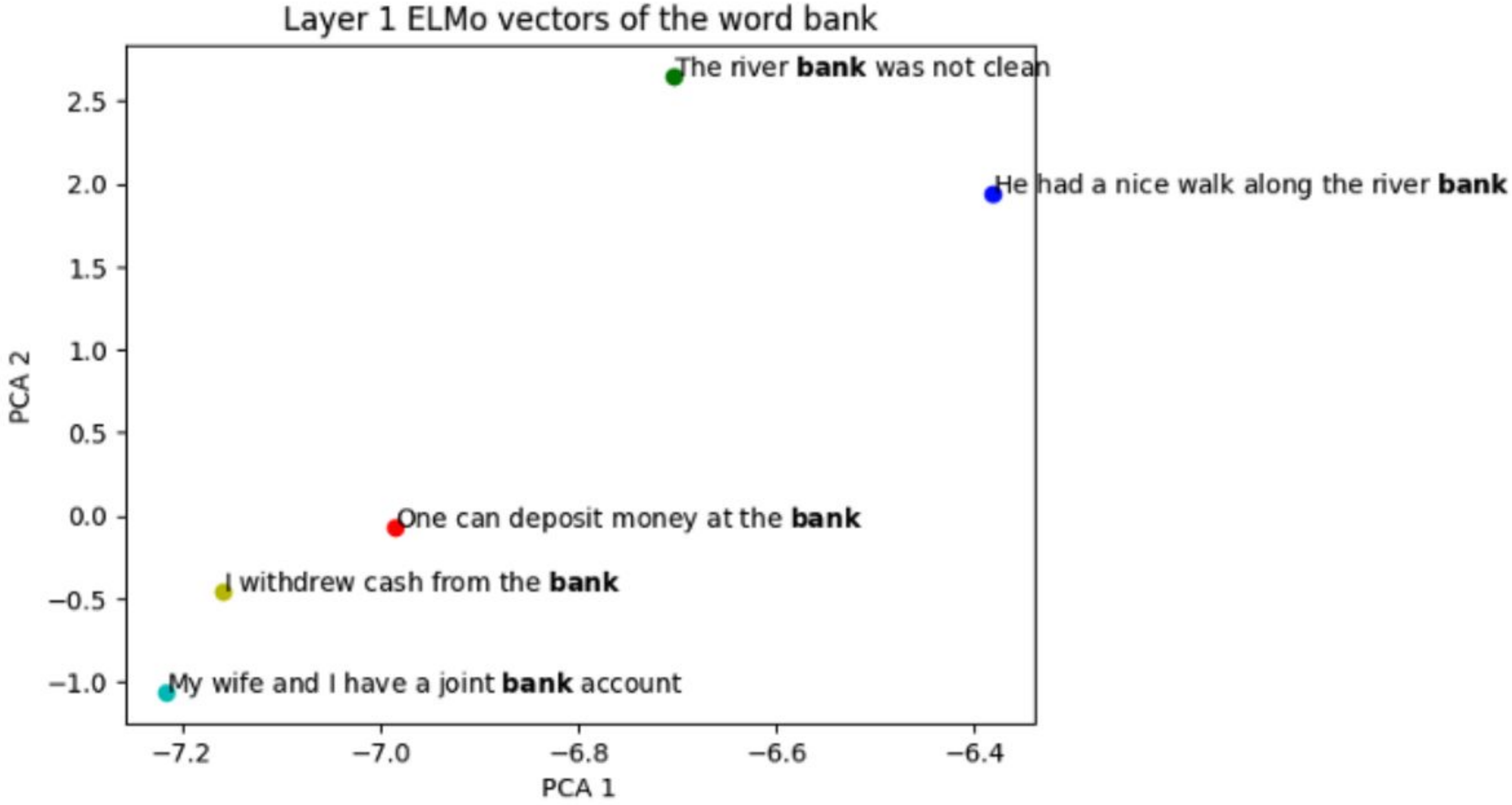
Coref: (average F1) coreference resolution.

NER: (F1) named entity resolution.

SST-5: (accuracy) sentiment analysis (5 labels).

Peters et al. "Deep contextualized word representations."

ELMo - Visualization



https://towardsdatascience.com/visualizing-elmo-contextual-vectors-94168768fdaa

Plan

- Natural Language Processing
- Words and Semantics
- Classical Approaches
- Word Embeddings
- Contextualized Word Embeddings
- **BERT**
- Summary

Recap

- Word embeddings provide efficient distributed representations for words.
  - They are based on an unsupervised pre-training phase that helps with scarce-data (supervised) tasks.

- ELMo and other similar approaches generate contextualized word embeddings that provide better results than uncontextualized embeddings.
  - Some of those approaches are based on an unsupervised pre-training phase. Others are based on a supervised pre-training phase.

- Do we have the best possible representation for words now?

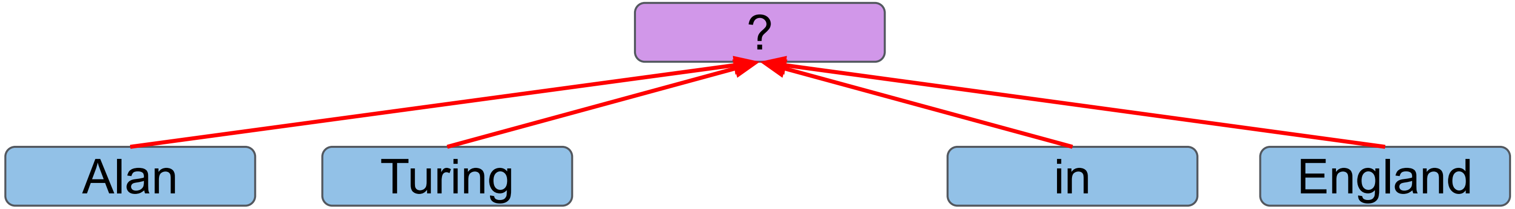
BERT

- Not yet!
- BERT (Bidirectional Encoder Representations from Transformers) improved results over many NLP tasks by:
  - using a Transformer-based architecture;
  - pre-training on two unsupervised tasks:
    - Masked Language Modeling (MLM) instead of Language Modeling;
    - Next sentence prediction.

Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

Masked Language Model

- Task: given a sentence where some tokens have been randomly masked, reconstruct the masked tokens.

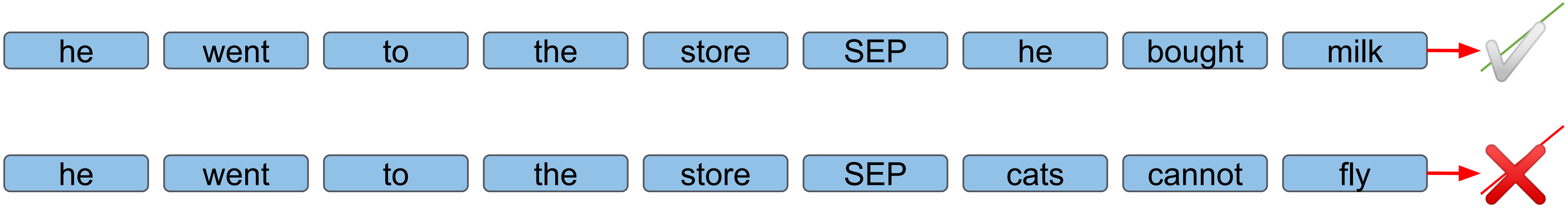


- Note there is no left-to-right or right-to-left order:
  - When predicting the missing word, the model has access to both the past and the future.

Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

Next Sentence Prediction

- Task: given two sentences, predict if they are contiguous.



he went to the store SEP he bought milk → ✓

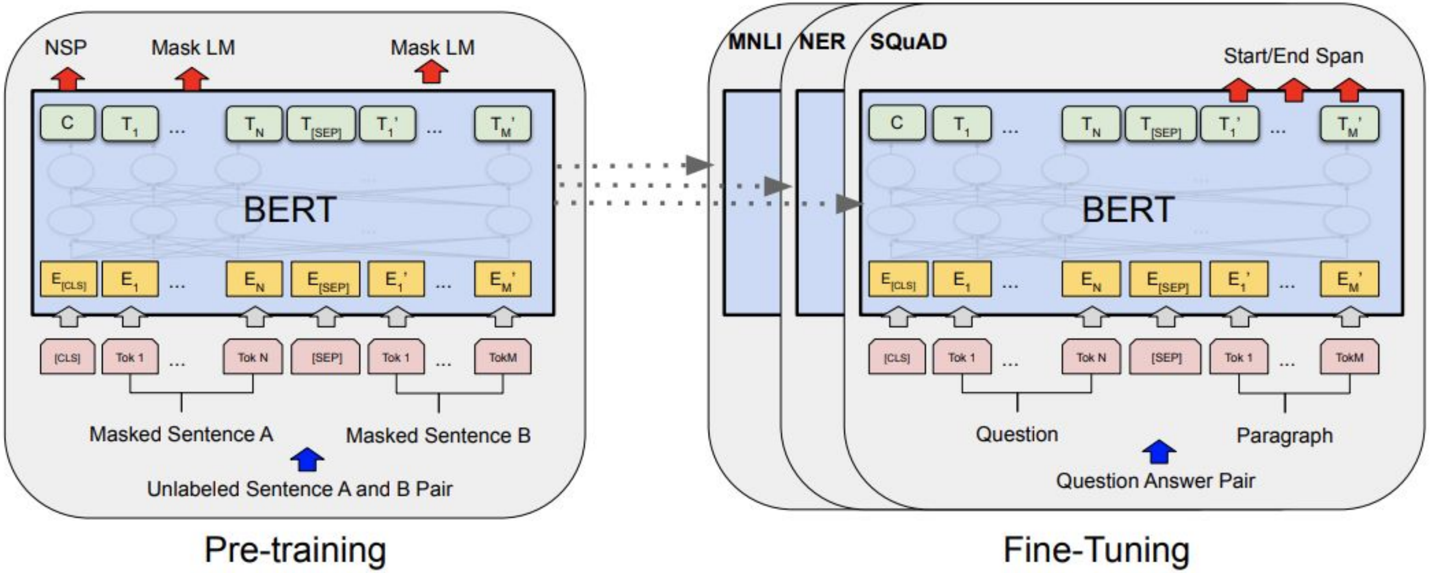
he went to the store SEP cats cannot fly → ✗

- Each contiguous example is created by extracting 2 successive sentences from a corpus.
- Each non contiguous example is created by selecting 2 random sentences from a corpus.
- This task helps to learn to capture relationships between sentences.
    - E.g., useful for Question Answering.

Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

BERT - Training Phase

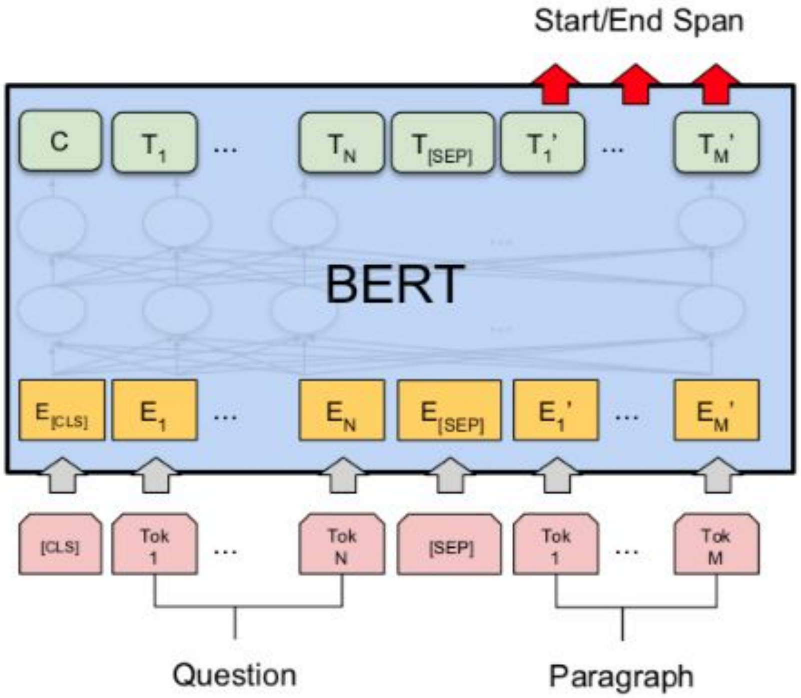
- After pre-training, the BERT model can be trained on the task of interest.
- Different kinds of tasks will require different input/output formalization.
  - Similar to what we saw in the NLP task section.

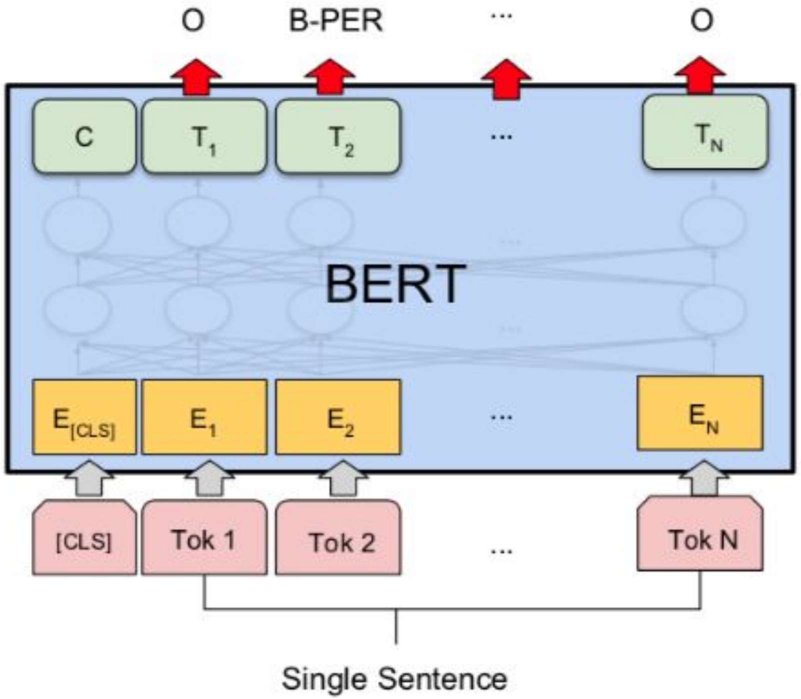


Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

BERT - Training Phase

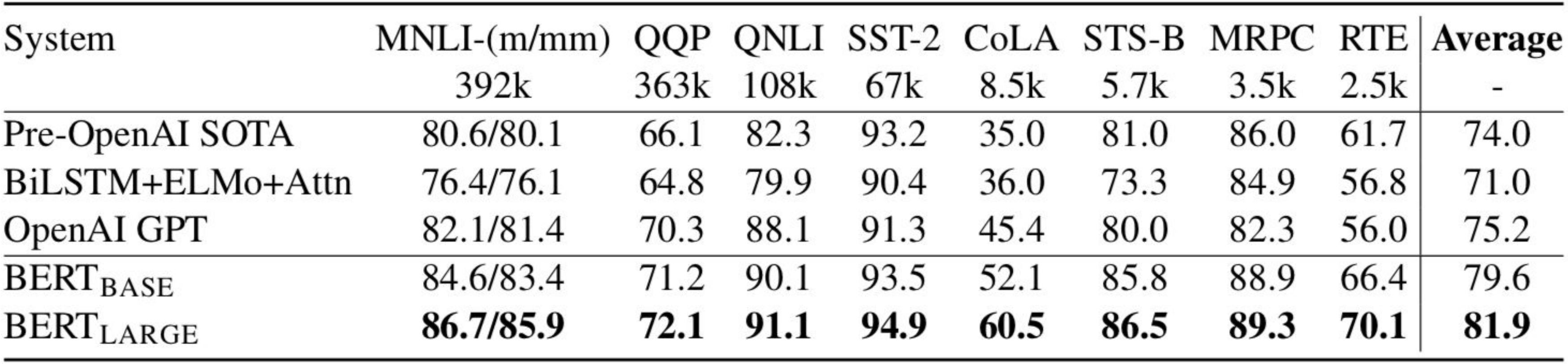
- E.g., Question Answering (left) and Named Entity Recognition (right):





Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

BERT - Results



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MNLI: (accuracy) sentence pair-level classification. (entailment)
QQP: (F1) sentence pair-level classification. (are these questions semantically equivalent?)
QNLI: (accuracy) sentence pair-level classification. (is the sentence an answer to the question?)
SST-2: (accuracy) sentence-level classification. (sentiment analysis - 2 labels).
CoLA: (custom metric) sentence-level classification. (is the sentence linguistically acceptable?)
STS-B: (custom metric) sentence pair-level classification. (are these sentences semantically equal? label 1 to 5)
MRPC: (F1) sentence pair-level classification. (are these sentences semantically equal? yes or no)
RTE: (accuracy) sentence pair-level classification. (entailment)

Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

Plan

- Natural Language Processing
- Words and Semantics
- Classical Approaches
- Word Embeddings
- Contextualized Word Embeddings
- BERT
- **Summary**

Summary

- Natural Language Processing includes many types of tasks.
- These tasks share some common problems such as the need to link words to semantics.
- Several algorithms have been developed to address these problems, mainly word embeddings (such as Word2Vec / FastText) and contextualized word embeddings (such as ELMo / BERT).

Summary

- BERT is able to address many NLP tasks with a common core architecture (based on the Transformer).
- All these algorithms (Word2Vec / FastText / ELMo / BERT) use the idea of (unsupervised) pre-training to help address scarcity of data for supervised tasks.
- New algorithms are created regularly!
    - E.g., XLNet, ERNIE, RoBERTa...

Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding."
Zhang et al. “ERNIE: Enhanced Language Representation with Informative Entities”
Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”

Questions?

ELMo - Pre-Training

- In pre-training, the goal is to maximize the following function:

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

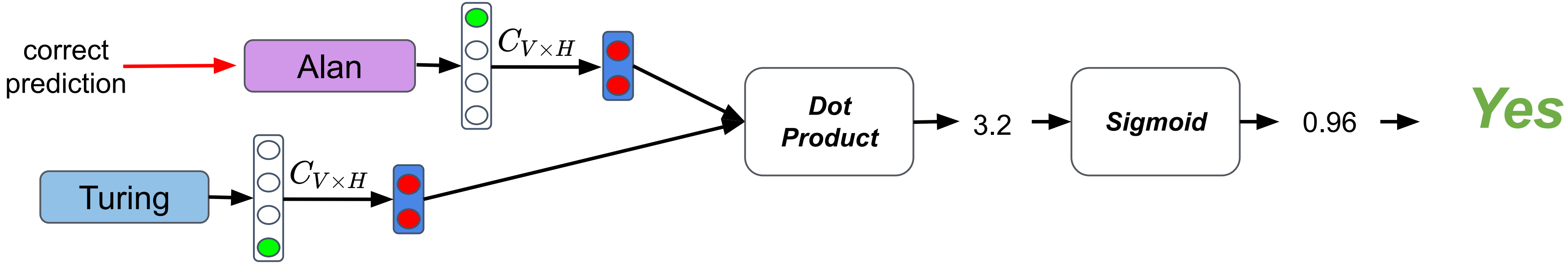
  where Θ_x and Θ_s are respectively the token representation and softmax layer parameters. Those parameters are shared by both LMs.

- Note that the hidden states of all layers are used to generated the contextualized embedding.

Peters et al. "Deep contextualized word representations."

Word2Vec - Negative Sampling

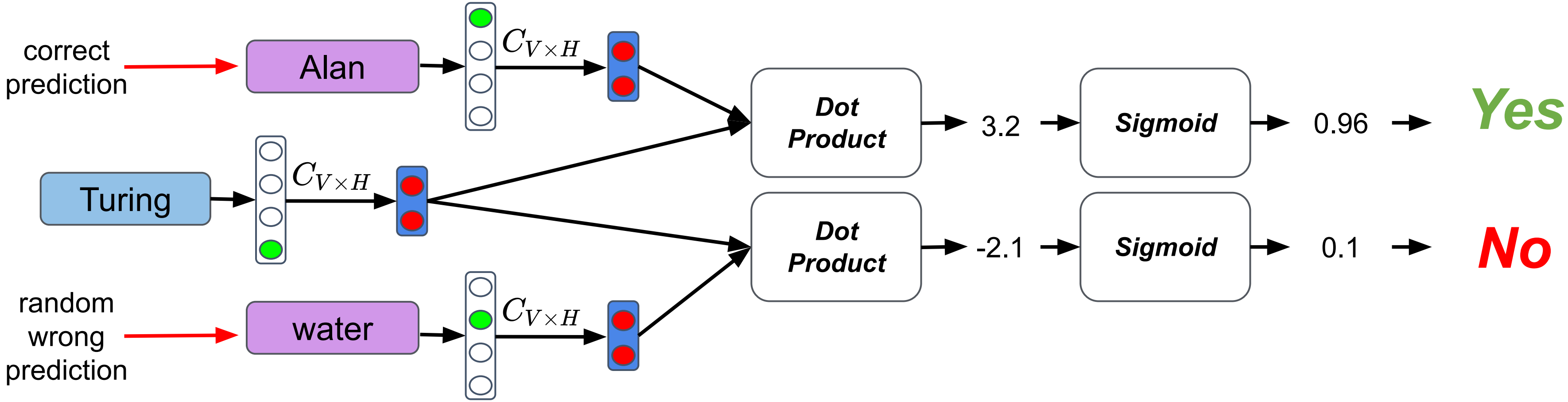
- Word2Vec may be expensive to train. Assuming *V* is the vocabulary size, the SoftMax is computed over *V* elements (which can be expensive if V is large).
- The SoftMax can be replaced with a Sigmoid computed only on K elements with K << V.
    - These elements include the correct prediction...



Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality."

Word2Vec - Negative Sampling

- Word2Vec may be expensive to train. Assuming *V* is the vocabulary size, the SoftMax is computed over *V* elements (which can be expensive if V is large).
- The SoftMax can be replaced with a Sigmoid computed only on K elements with K << V.
  - These elements include the correct prediction…
  - … plus K-1 wrong predictions (sampled proportionally to word frequencies).

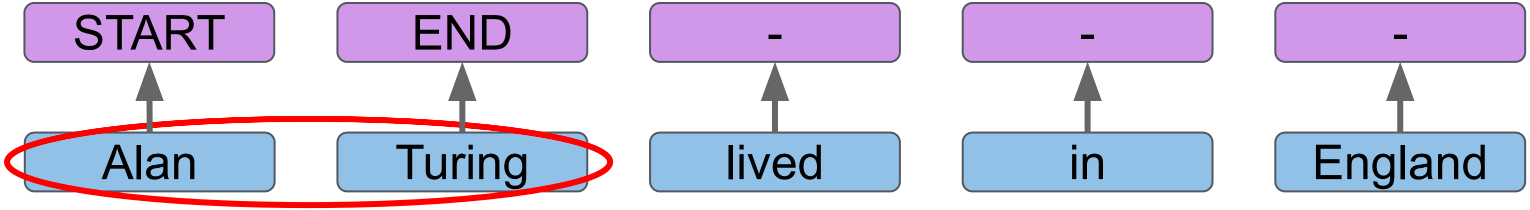


NLP - Extractive Question Answering

- Task: given a question, find the answer in some given text. E.g.,
  - (Input) Question: “*who was living in England?*”
  - (Input) Context: “*Alan Turing lived in England*”
  - Target: “*Alan Turing*” (i.e., first and second words in the context)

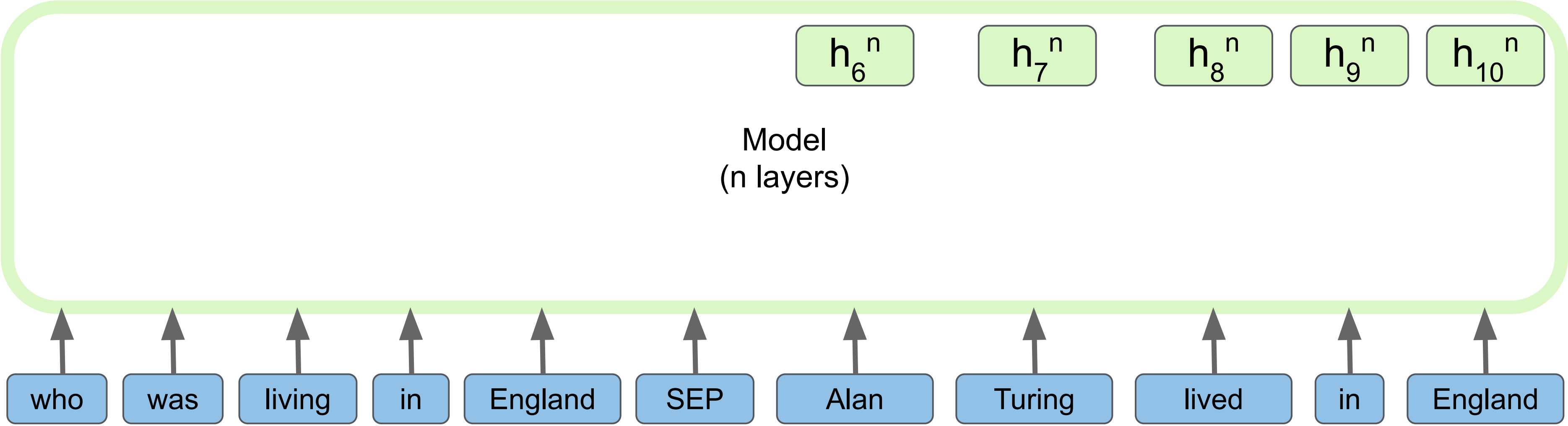
NLP - Extractive Question Answering

- Task: given a question, find the answer in some given text. E.g.,
  - (Input) Question: "*who was living in England?*"
  - (Input) Context: "*Alan Turing lived in England*"
  - Target: "*Alan Turing*" (i.e., first and second words in the context)
- This task can be modeled as a word-level classification task.
  - Each word can be the start of the answer, the end of the answer, or neither of them.

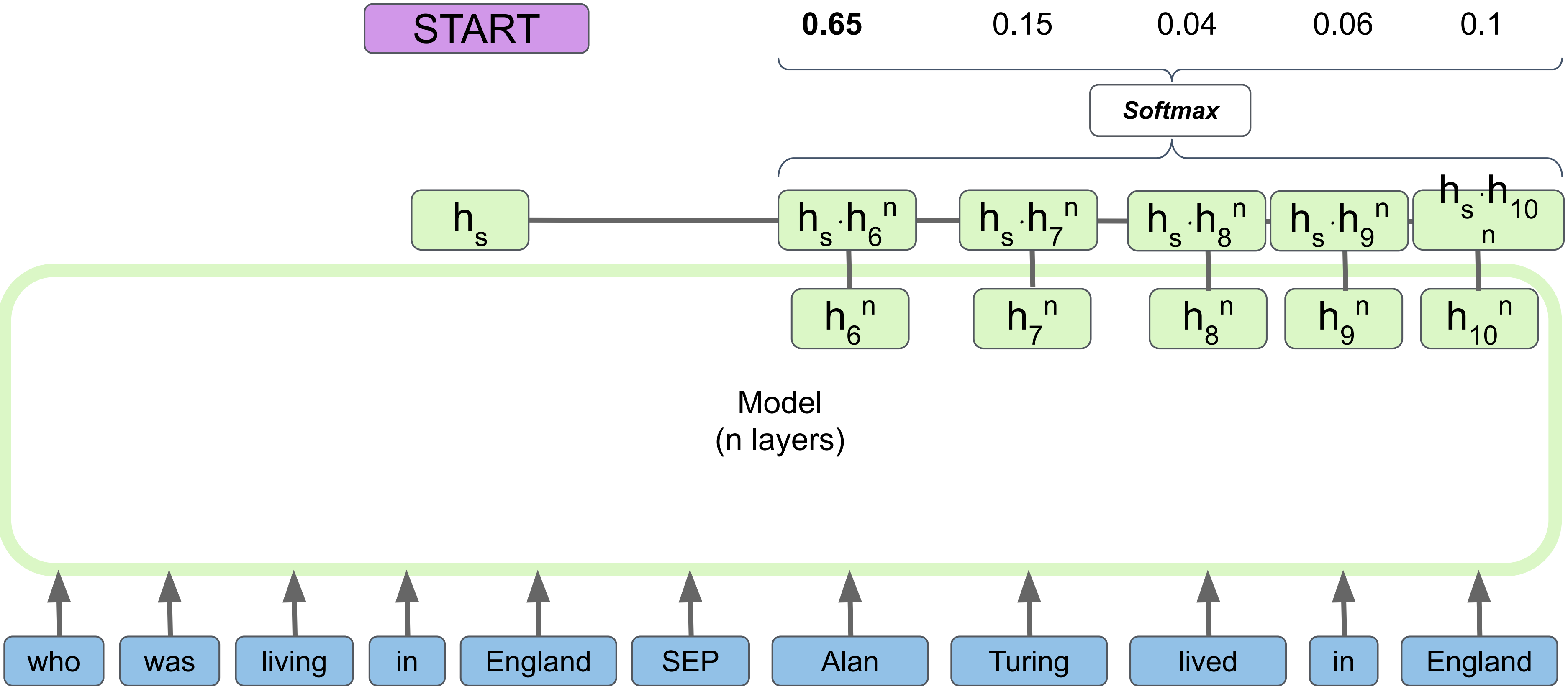


NLP - Extractive Question Answering

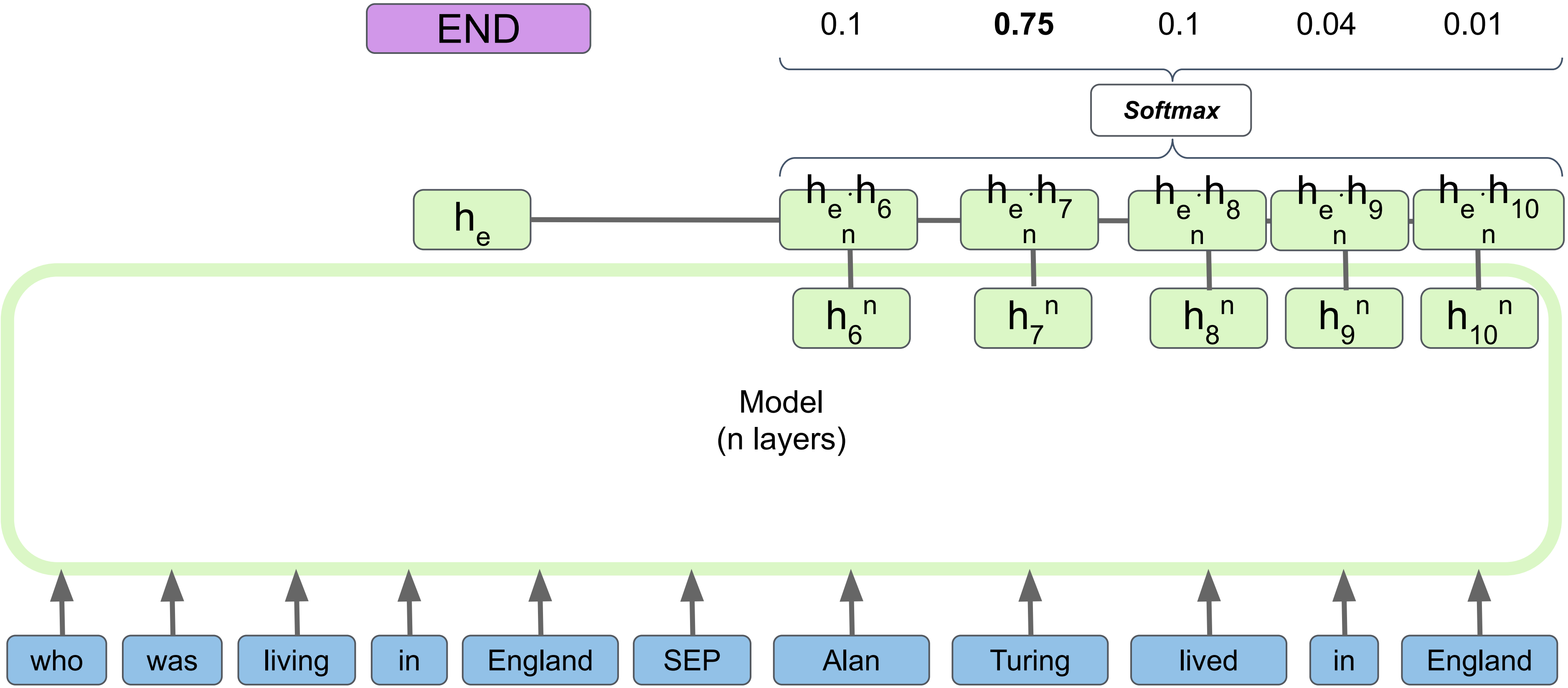
SEP = separator
indicating the end of
the first sentence.



NLP - Extractive Question Answering



NLP - Extractive Question Answering


END
0.1
0.75
0.1
0.04
0.01
Softmax
h_e
$h_e \cdot h_6^n$
$h_e \cdot h_7^n$
$h_e \cdot h_8^n$
$h_e \cdot h_9^n$
$h_e \cdot h_{10}^n$
h_6^n
h_7^n
h_8^n
h_9^n
h_{10}^n
Model
(n layers)
who
was
living
in
England
SEP
Alan
Turing
lived
in
England