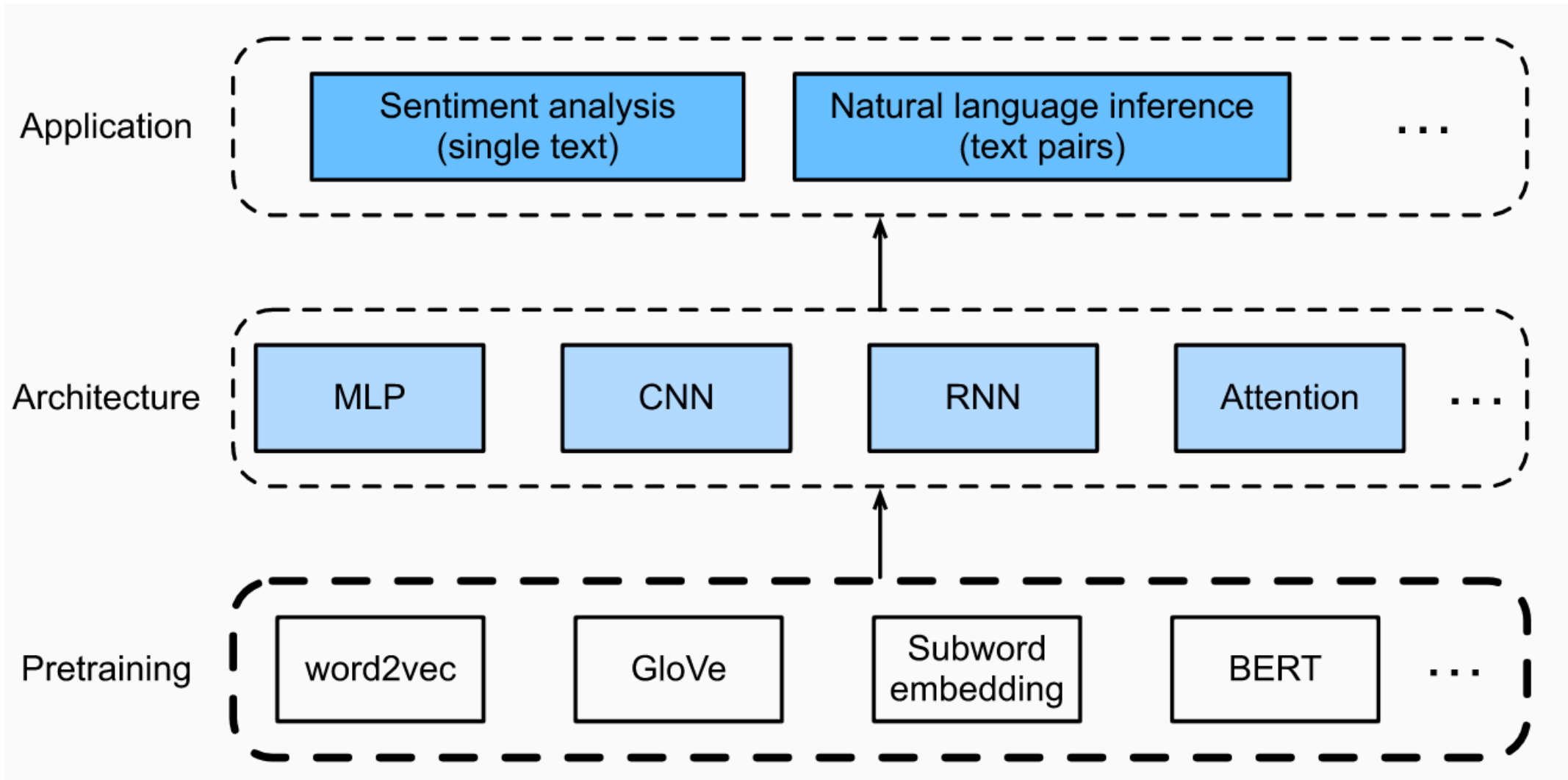


NLP – Part 2

Arsalan Mosenia (am12546@nyu.edu)

Chinmay Hegde (chinmay.h@nyu.edu)

NLP Overview



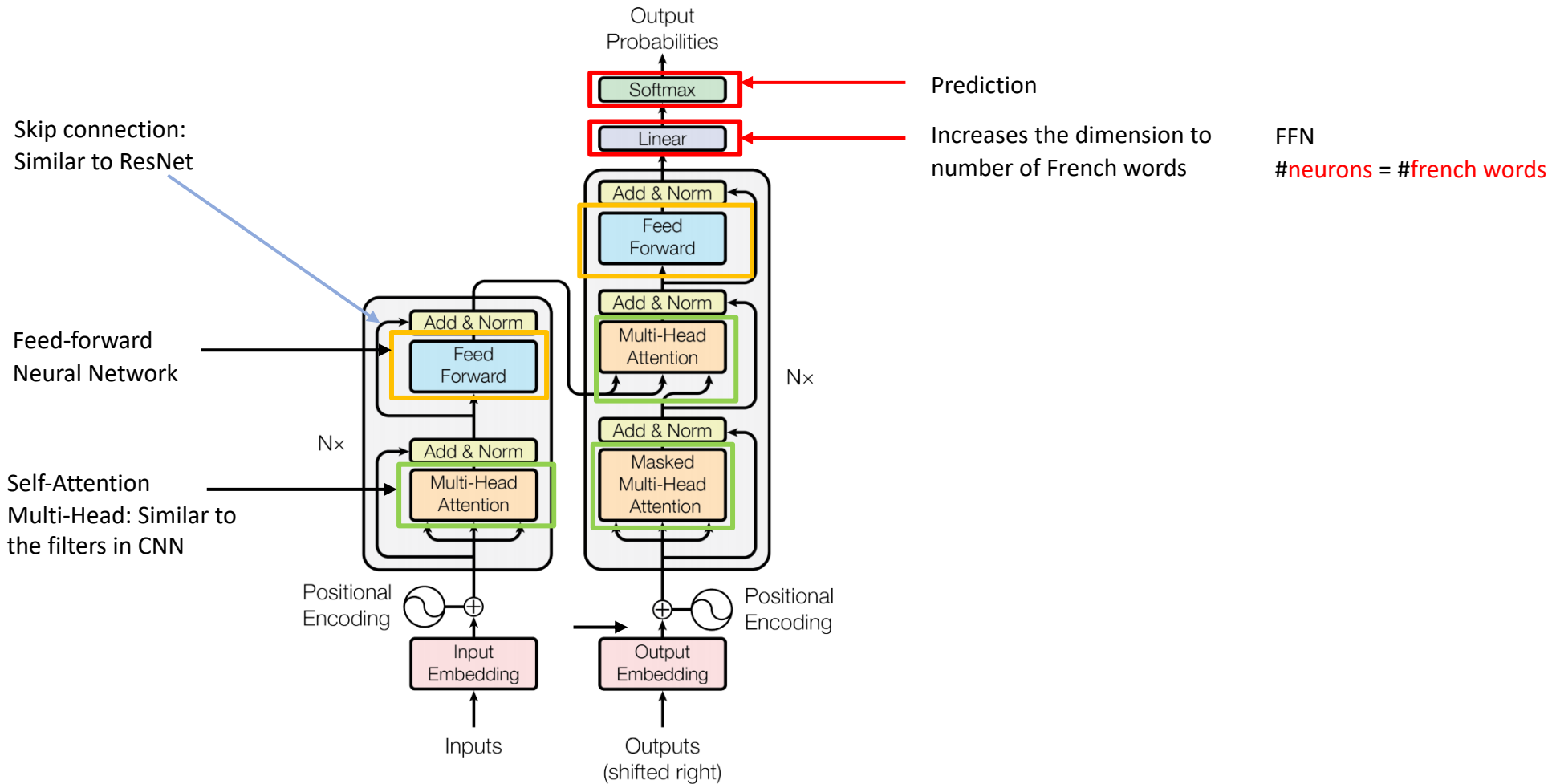
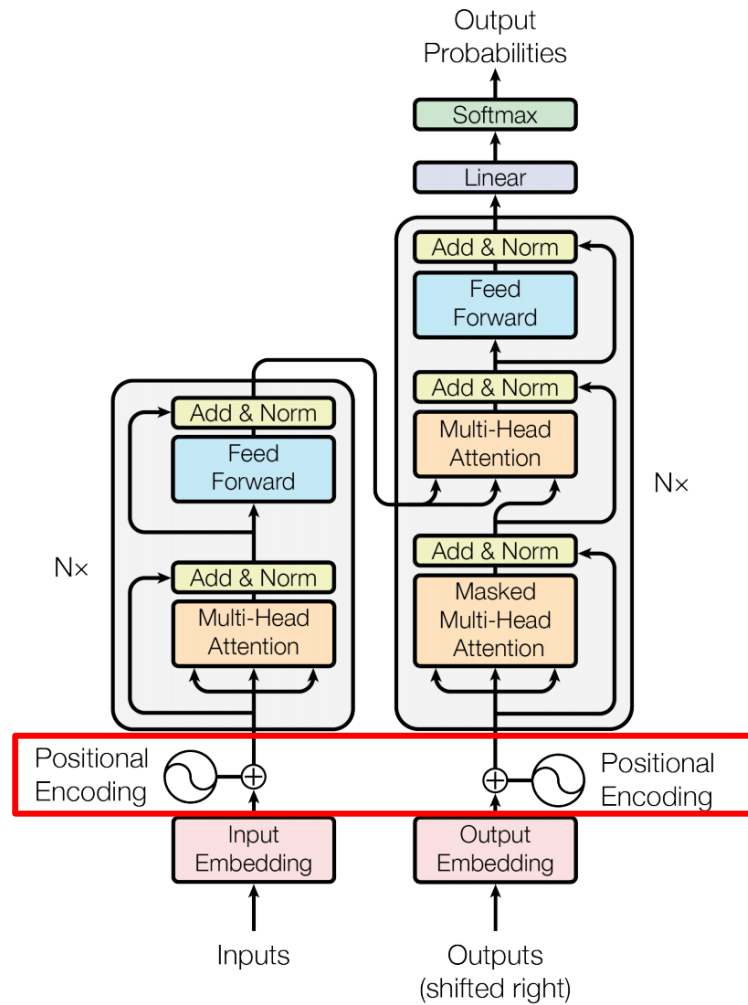


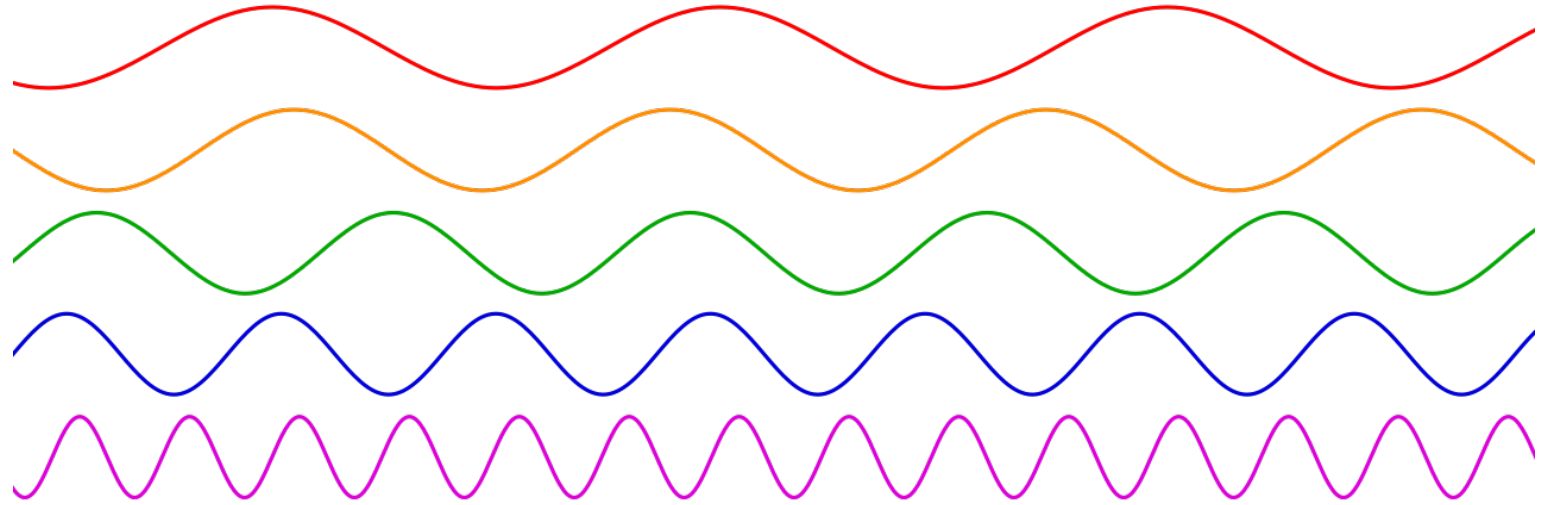
Figure 1: The Transformer - model architecture.



Positional Embedding

Positional Encoding

Use Sine or Cosine with different frequencies.



Positional Encoding

$$PE(Pos, 2i) = \sin\left(\frac{pos}{N^{\frac{2i}{d_{model}}}}\right)$$

$$PE(Pos, 2i + 1) = \cos\left(\frac{pos}{N^{\frac{2i}{d_{model}}}}\right)$$

$N = A$ large number

d = size of word vectors

$$PE(Pos, 2i + 1) = \cos\left(\frac{pos}{100^{\frac{2i}{4}}}\right) \quad PE(Pos, 2i) = \sin\left(\frac{pos}{100^{\frac{2i}{4}}}\right)$$

Sequence	Index of token	Positional Encoding Matrix			
I	0	P_{00}	P_{01}	...	P_{0d}
am	1	P_{10}	P_{11}	...	P_{1d}
a	2	P_{20}	P_{21}	...	P_{2d}
Robot	3	P_{30}	P_{31}	...	P_{3d}

Positional Encoding Matrix for the sequence 'I am a robot'

Positional Encoding

$$PE(Pos, 2i + 1) = \cos\left(\frac{pos}{100^{\frac{2i}{4}}}\right)$$

$$PE(Pos, 2i) = \sin\left(\frac{pos}{100^{\frac{2i}{4}}}\right)$$

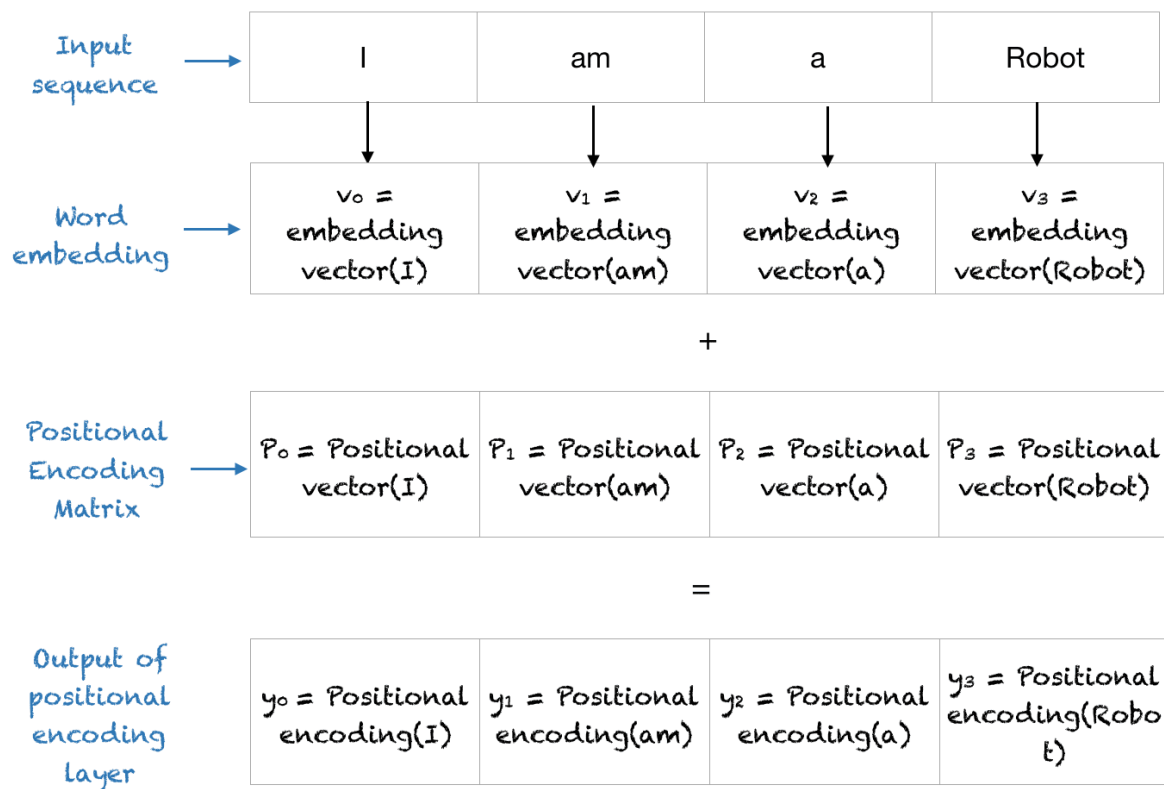
Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	$P_{00}=\sin(0)$ = 0	$P_{01}=\cos(0)$ = 1	$P_{02}=\sin(0)$ = 0	$P_{03}=\cos(0)$ = 1
am	1	$P_{10}=\sin(1/1)$ = 0.84	$P_{11}=\cos(1/1)$ = 0.54	$P_{12}=\sin(1/10)$ = 0.10	$P_{13}=\cos(1/10)$ = 1.0
a	2	$P_{20}=\sin(2/1)$ = 0.91	$P_{21}=\cos(2/1)$ = -0.42	$P_{22}=\sin(2/10)$ = 0.20	$P_{23}=\cos(2/10)$ = 0.98
Robot	3	$P_{30}=\sin(3/1)$ = 0.14	$P_{31}=\cos(3/1)$ = -0.99	$P_{32}=\sin(3/10)$ = 0.30	$P_{33}=\cos(3/10)$ = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

Positional Encoding

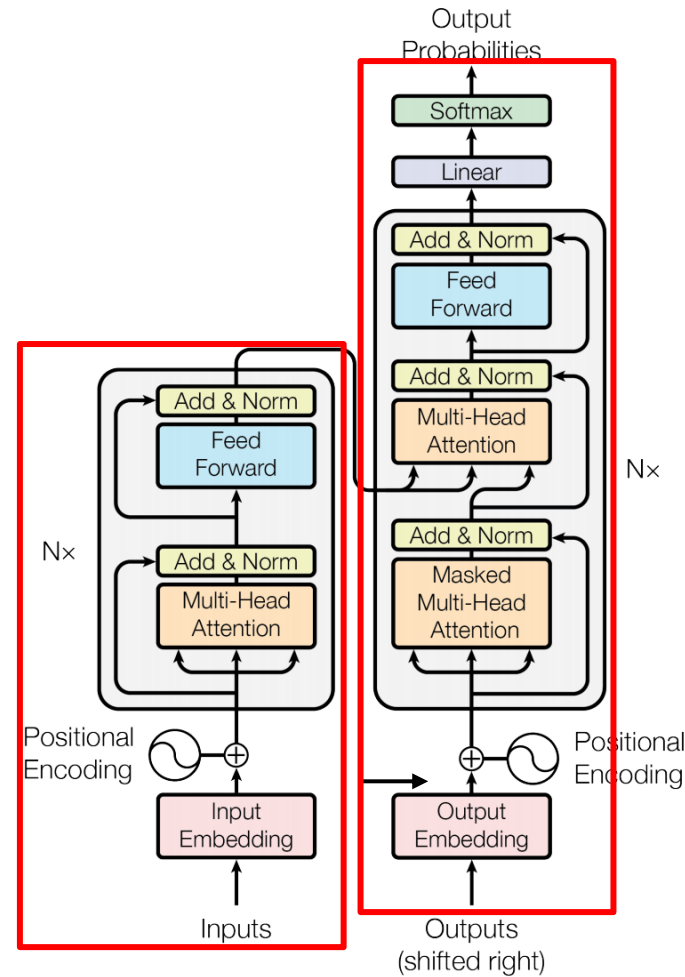
Transformers:

- Use Sine/Cosine for PE
- Without PE, the input is bag of words
- We can potentially use Sine only
- This is one way of implementing this, but there is not standard way to inject positional information.
- In practice, $d = 512$, and $N = 10000$



Summary

What is English?
What is Language?
What is context?



What is French?
How to map English to French?
What is Language?
What is context?

Figure 1: The Transformer - model architecture.

Word Embeddings

One Hot Encoding

- Each word in the vocabulary is represented by one bit position in a HUGE vector.
- For example, if we have a vocabulary of 10000 words, and “Air” is the 4th word in the dictionary, it would be represented by:

Air = [0 0 0 1 0 0 0 0 0]

- Wasting large v-dimensional space
- Context information is not utilized.

Word Embeddings

- Stores each word in as a point in space, where it is represented by a dense vector of fixed number of dimensions (e.g., 300).
- For example, “Air” might be represented as:

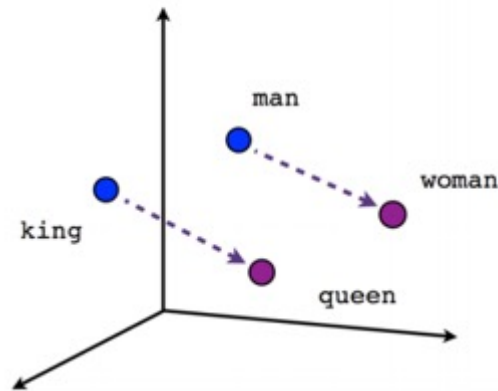
Air = [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]

- Dimensions are basically projections along different axes, more of a mathematical concept.
- Built just by reading huge corpus.

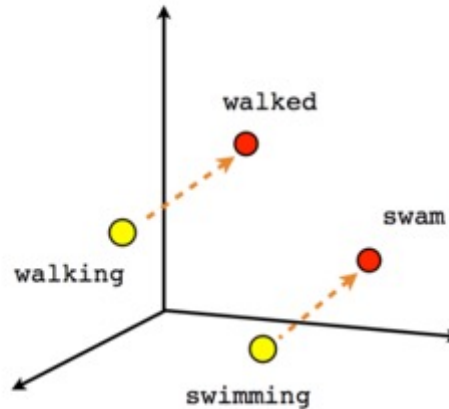
Different Word Embeddings

- Latent Semantic Analysis/Indexing (1988)
 - Term weighting-based model
 - Consider occurrences of terms at document level.
- Word2Vec (2013)
 - Prediction-based model.
 - Consider occurrences of terms at context level.
- GloVe (2014)
 - Count-based model.
 - Consider occurrences of terms at context level.
- ELMo (2018)
 - Language model-based.
 - A different embedding for each word for each task.
- BERT (2018)

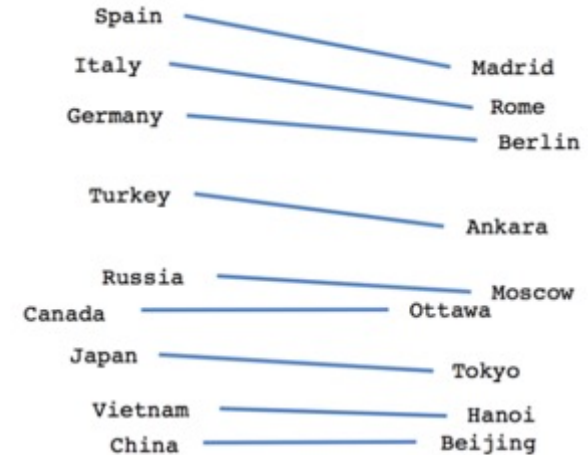
Relationships



Male-Female



Verb tense



Country-Capital

- $E[\text{Queen}] \approx E[\text{King}] - E[\text{Man}] + E[\text{Woman}]$
- $E[\text{Paris}] \approx E[\text{France}] - E[\text{Italy}] + E[\text{Rome}]$
 - This can be interpreted as “France is to Paris as Italy is to Rome”.

Vector Embedding of Words

- A word is represented as a **vector**.
- Word embeddings depend on a notion of **word similarity**.
 - Similarity is computed using cosine.
- A very useful definition is paradigmatic similarity:
 - **Similar words** occur in **similar contexts**. They are **exchangeable**.
- Yesterday

{	POTUS	}
	The President	
	Biden	

 called a press conference.
- “POTUS: President of the United States.”

Word2Vec

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov

Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen

Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado

Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean

Google Inc., Mountain View, CA
jeff@google.com

Abstract

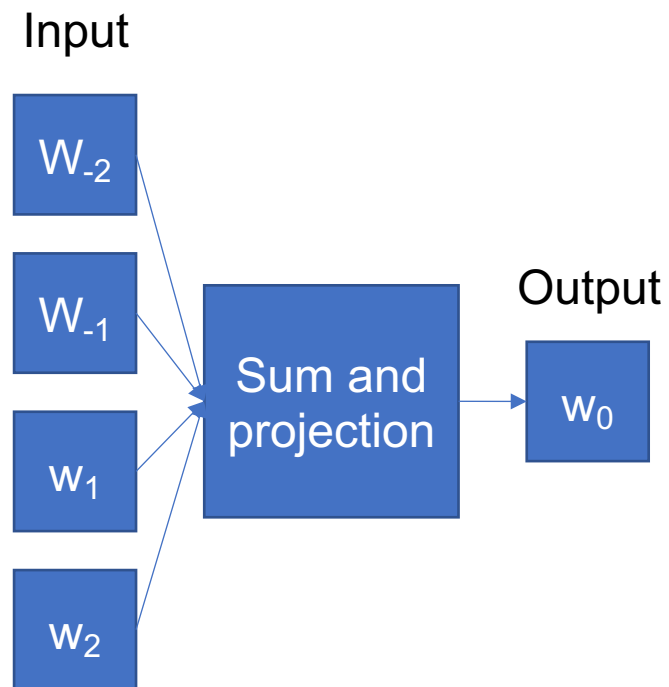
We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Word2Vec: Local Context

- **Word2Vec** uses words k positions away from each center word.
 - These words are called **context words**.
- Example for $k=2$:
 - “It was a bright cold day in April, and the clocks were striking”.
 - Center word: red (also called focus word).
 - Context words: blue (also called target words).
- Word2Vec considers all words as center words, and all their context words.

Word2Vec: Continuous Bag of Words

Continuous Bag of Words



Given a surrounding words can we predict the center word?

I have a black car, and she has a --- car.

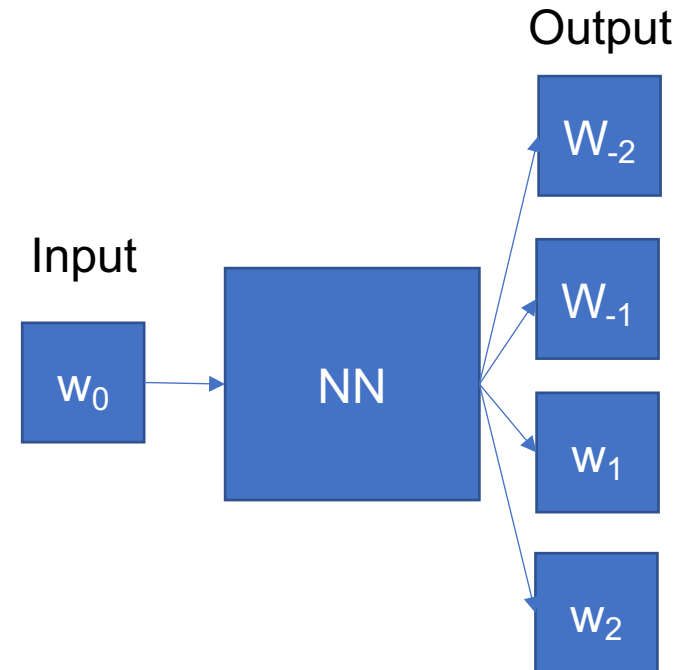
- Will focus on Skip-Gram model – Skip-Gram works better in practice.

Word2Vec: Skip-Gram

Skip-Gram Model

- Sliding window of a fixed size moving along a sentence.
- The word in the middle is the “target” and those on its left and right within the sliding window are the context words.
- The skip-gram model predicts the probabilities of a word being a context word for the given target.

Sentence: I have a **black** car, and she has a red car.



Word2Vec: Skip-Gram

“The man who passes the sentence
should swing the sword.”

Sliding window (size = 5)	Target word	Context
[The man who]	the	man, who
[The man who passes]	man	the, who, passes
[The man who passes the]	who	the, man, passes, the
[man who passes the sentence]	passes	man, who, the, sentence
...
[sentence should swing the sword]	swing	sentence, should, the, sword
[should swing the sword]	the	should, swing, sword
[swing the sword]	sword	swing, the

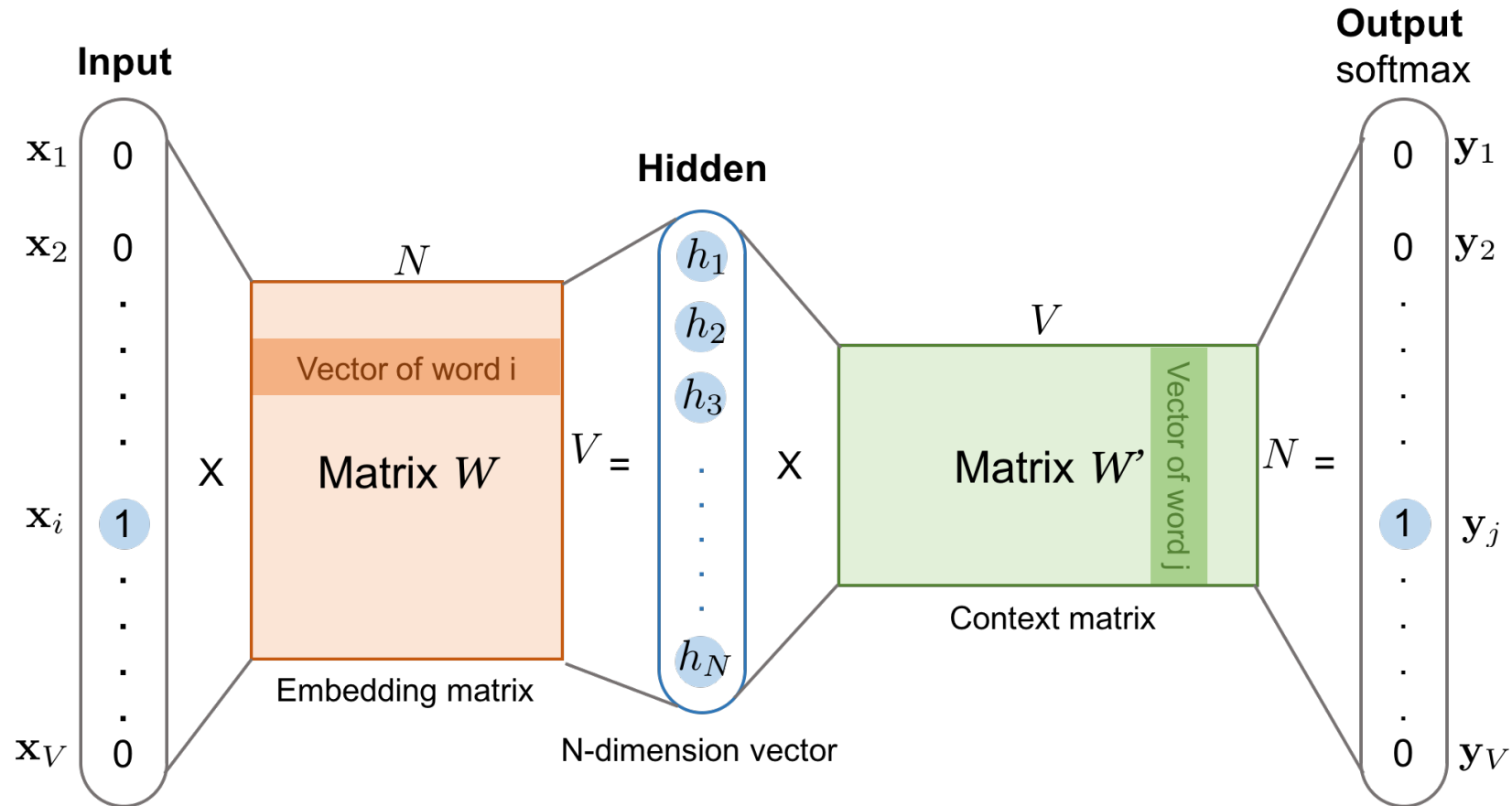
Word2Vec: Skip-Gram

“The man who passes the sentence should swing the sword.”

Each context-target pair is treated as a new observation in the data. For example, the target word “swing” in the above case produces four training samples: (“swing”, “sentence”), (“swing”, “should”), (“swing”, “the”), and (“swing”, “sword”)

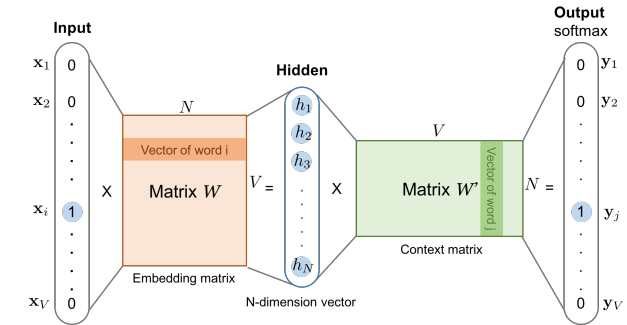
Sliding window (size = 5)	Target word	Context
[The man who]	the	man, who
[The man who passes]	man	the, who, passes
[The man who passes the]	who	the, man, passes, the
[man who passes the sentence]	passes	man, who, the, sentence
...
[sentence should swing the sword]	swing	sentence, should, the, sword
[should swing the sword]	the	should, swing, sword
[swing the sword]	sword	swing, the

Word2Vec: Skip-Gram



Word2Vec: Local Context

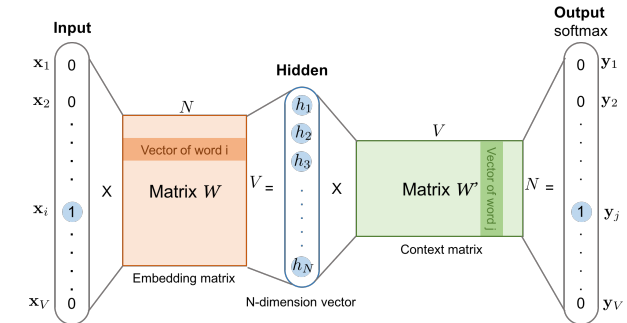
- Both input word and the output word are one-hot encoded into binary vectors X and Y of size $V \times 1$.
- First, the multiplication of the binary vector x and the word embedding matrix W of size $V \times N$ gives us the embedding vector of the input word X : The i -th row of the matrix W .
- This newly discovered embedding vector of dimension N forms the hidden layer.
- The multiplication of the hidden layer and the word context matrix W' of size $N \times V$ produces the output one-hot encoded vector Y .
- The output context matrix W' encodes the meanings of words as context, different from the embedding matrix W .



NOTE: Despite the name, W' is independent of W , not a transpose or inverse or whatsoever.

Summary

- The model learns the mapping so that it can predict context words given a focus word.
- Word2Vec learns low-dimensional vectorized representation for each word.
- After the training, we have a mapping from the V -dimensional vocabulary space to the d -dimensional embedding space.
- Note that if you give the one-hot encoding to your model, the hidden layer receives a low-dimensional space that is indeed the word embedding.



Word2Vec: Relationships

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Context is Key

- Language is complex, and **context** can completely change the meaning of a word in a sentence.
- Example:
 - I let the kids outside to **play**.
 - He had never acted in a more famous **play** before.
 - It wasn't a **play** the coach would approve of.
- Need a model which captures the different nuances of the meaning of words given the surrounding text.

Vector Embedding of Words

Non-contextualized Embeddings

- Earlier models (Word2Vec, etc) only have one representation per word
- They can not capture these ambiguities.
- When you only have one representation, all levels of meaning are combined.
- You fetch them from a look-up table

Contextualized Word Embeddings

- Take context into account
- Examples are ELMO and BERT
- One word can have multiple presentations
- You can feed Non-contextualized embeddings to models (e.g., Transformers) to create contextualized embeddings

"After stealing money from the bank vault, the bank robber was seen
"fishing on the Mississippi river bank."

Understanding Language/Context

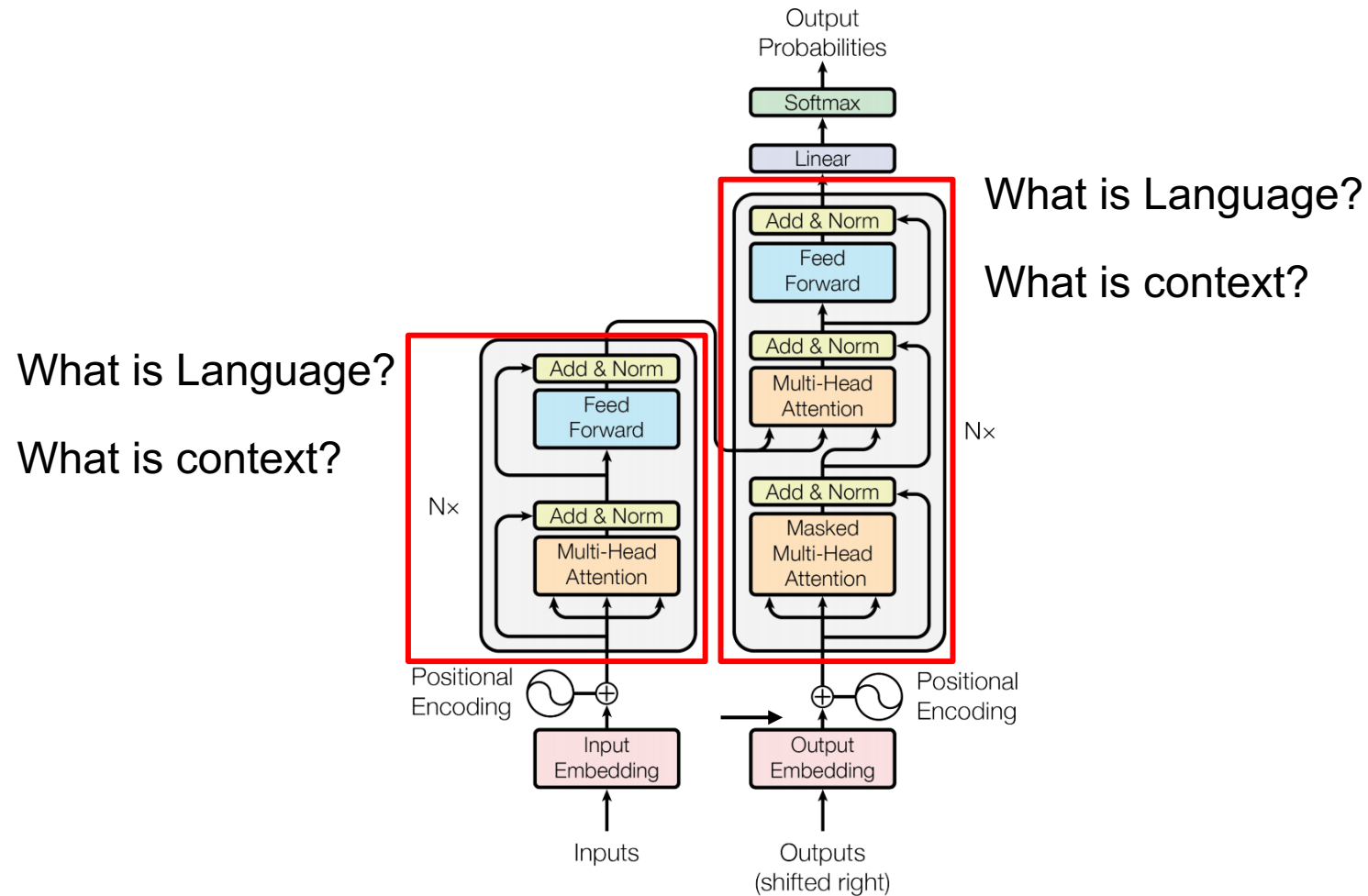
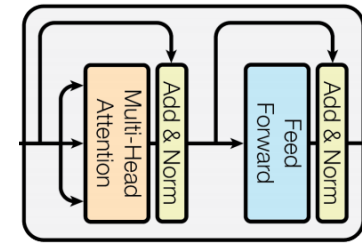
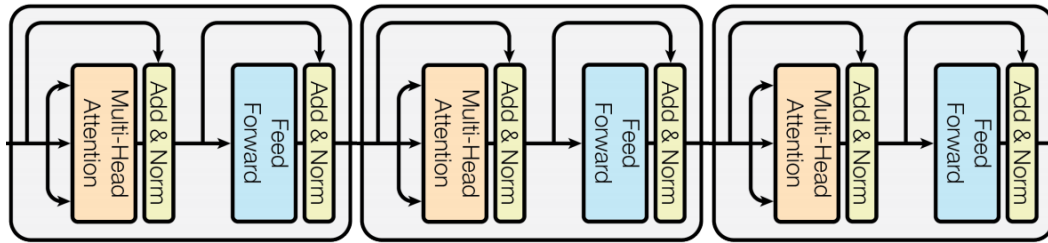


Figure 1: The Transformer - model architecture.

Bert

Bert



Pre-train Bert

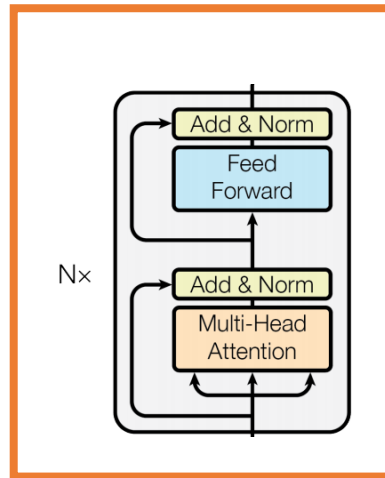
We can pre-train Bert to understand the language and context:

- Masked Language Model (MLM)
- Next Sentence Prediction (NSP)

The [MASK1] is blue but
trees are [MASK]

A = DP is a good course.
B = It does not have a final exam.

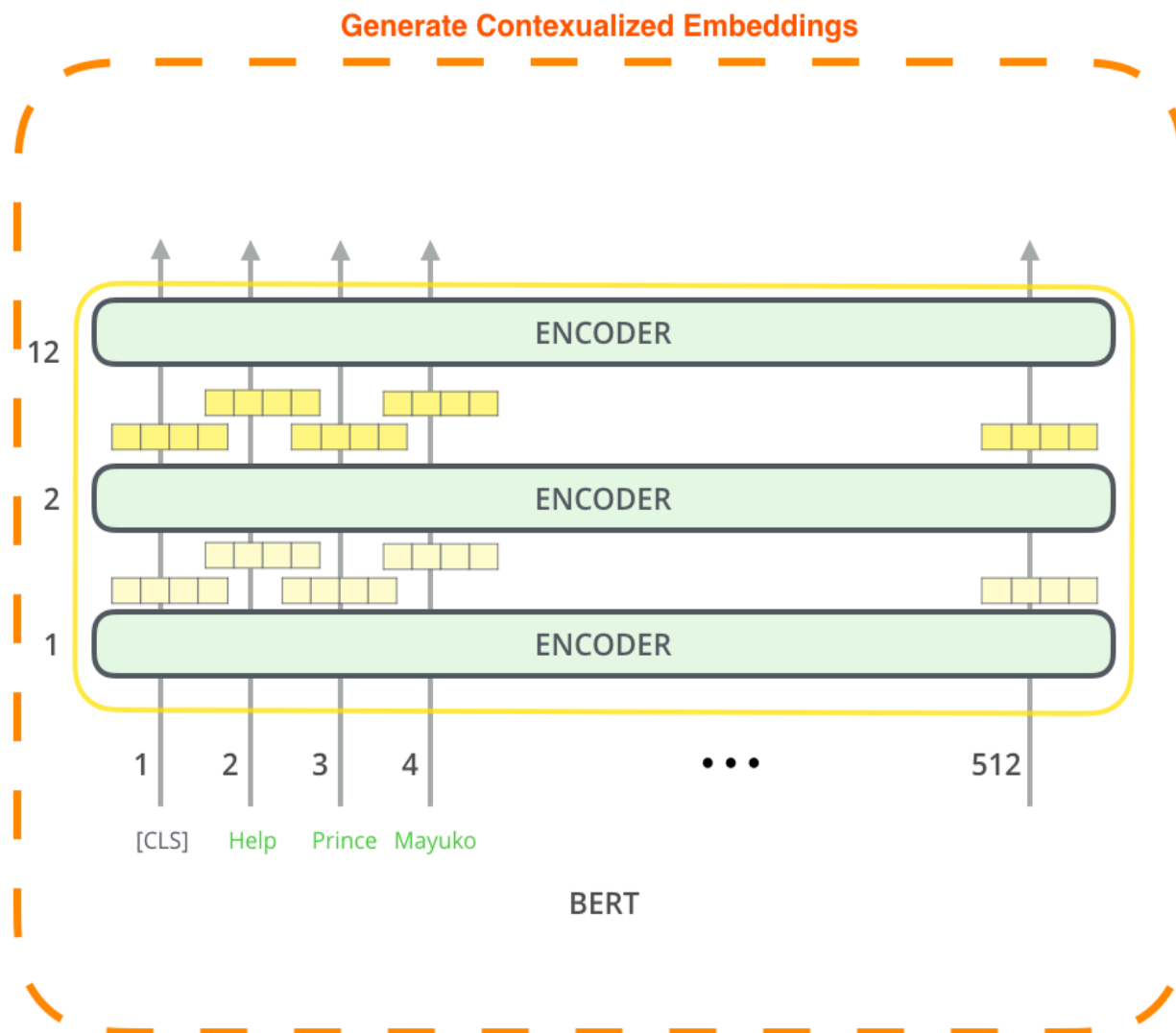
Bert



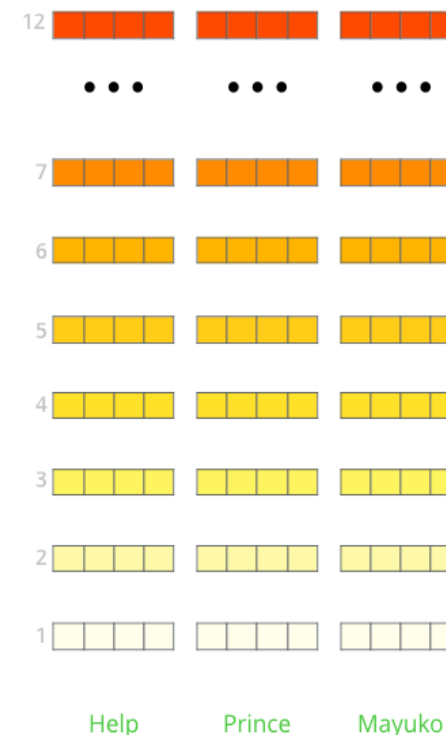
Mask1 = sky
Mask2 = green

Yes. Sentence B follows A

Bert for Contextualized Word Embeddings



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

How to combine hidden values?

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER

12 

...

7 

6 

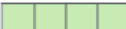
5 

4 

3 

2 

1 



Help

First Layer

Embedding 

Last Hidden Layer

12 

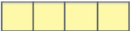
Sum All 12
Layers

12 

+

...

+

2 

+

1 

=



Second-to-Last
Hidden Layer

11 

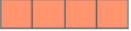
Sum Last Four
Hidden

12 

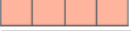
+

11 

+

10 


+

9 

=



Concat Last
Four Hidden



Example: Contextualized Word Embedding

"After stealing money from the bank vault, the bank robber was seen
"fishing on the Mississippi river bank."

Strategy: Summing the last four layers

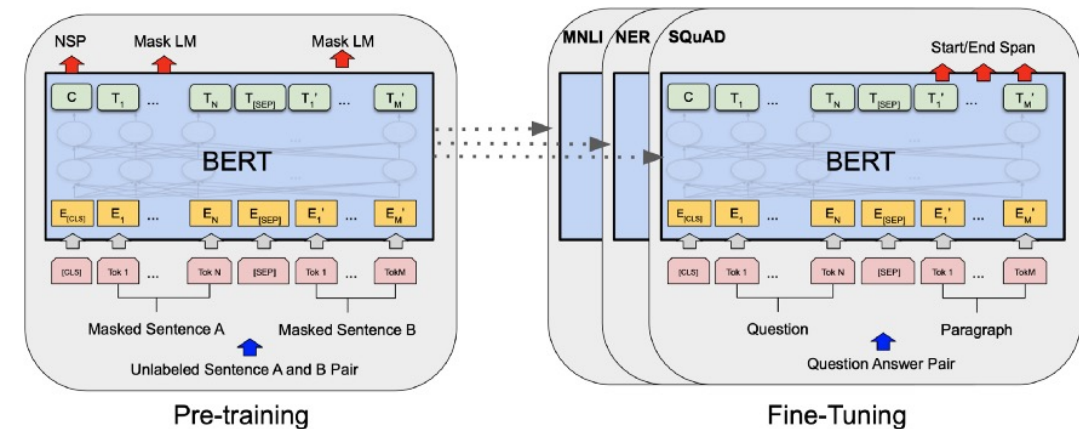
"Bank" vault tensor([3.3596, -2.9805, -1.5421, 0.7065, 2.0031])

"Bank" robber tensor([2.7359, -2.5577, -1.3094, 0.6797, 1.6633])

"River" bank tensor([1.5266, -0.8895, -0.5152, -0.9298, 2.8334])

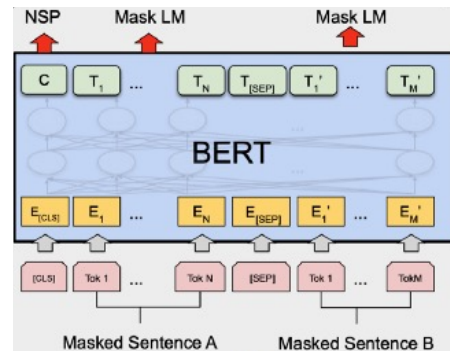
Transfer Learning: Bert

1. Sentiment analysis: Example: Is a review good/bad?
2. Question Answering: Receives a question regarding a text sequence and is required to mark the answer in the sequence.
3. In Named Entity Recognition (NER): Marks the various types of entities (Person, Organization, Date, etc) that appear in the text.

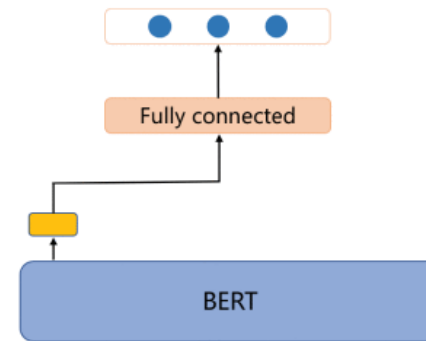


Transfer Learning: Bert

Sentiment Analysis with Bert: Add a fully-connected layer to the first output.
Intuition: C has some overall understanding of the whole input



Positive/Negative/Neutral



BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis
[Hu Xu](#), [Bing Liu](#), [Lei Shu](#), [Philip S. Yu](#)

Dual-Encoder

StarSpace: Embed All The Things!

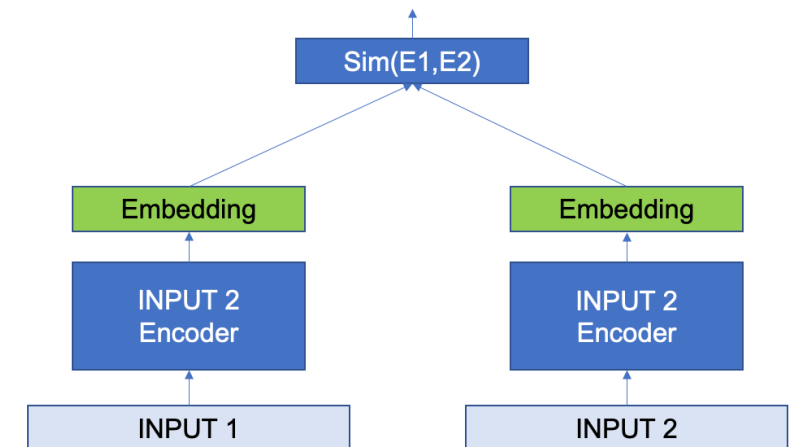
Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes and Jason Weston
Facebook AI Research

Abstract

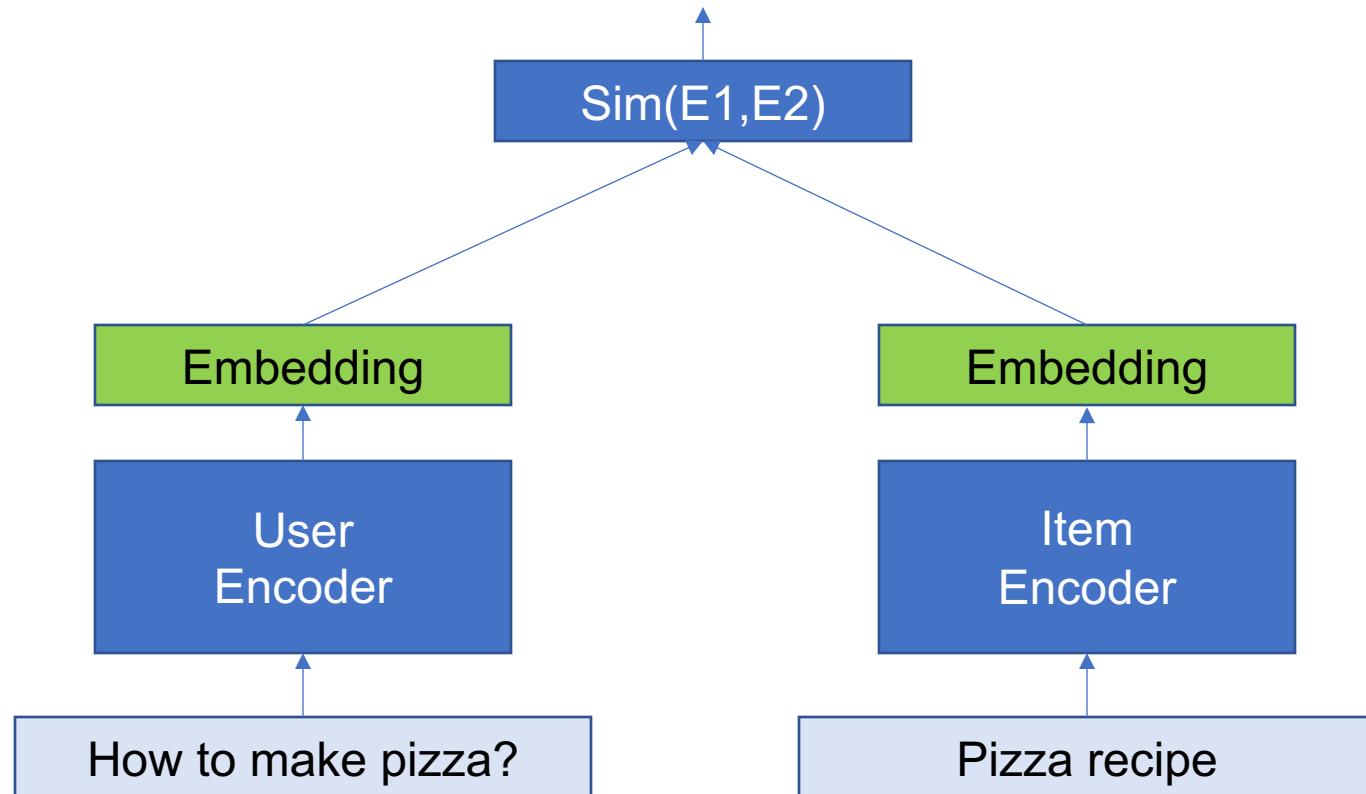
We present StarSpace, a general-purpose neural embedding model that can solve a wide variety of problems: labeling tasks such as text classification, ranking tasks such as information retrieval/web search, collaborative filtering-based or content-based recommendation, embedding of multi-relational graphs, and learning word, sentence or document level embeddings. In each case the model works by embedding those entities comprised of discrete features and comparing them against each other – learning similarities dependent on the task. Empirical results on a number of tasks show that StarSpace is highly competitive with existing methods, whilst also being generally applicable to new cases where those methods are not.

hence the “star” (“*”, meaning all types) and “space” in the name, and in that common space compares them against each other. It learns to rank a set of entities, documents or objects given a query entity, document or object, where the query is not necessarily of the same type as the items in the set.

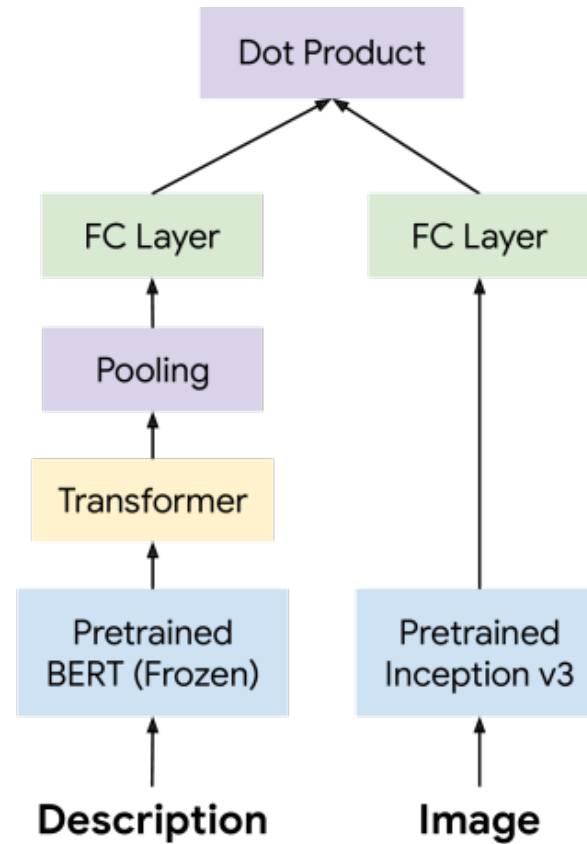
We evaluate the quality of our approach on six different tasks, namely text classification, link prediction in knowledge bases, document recommendation, article search, sentence matching and learning general sentence embeddings. StarSpace is available as an open-source project at <https://github.com/facebookresearch/Starspace>.



Dual-Encoder for Search



Dual-Encoder for Image Search



Modern RecSys in a Nutshell

