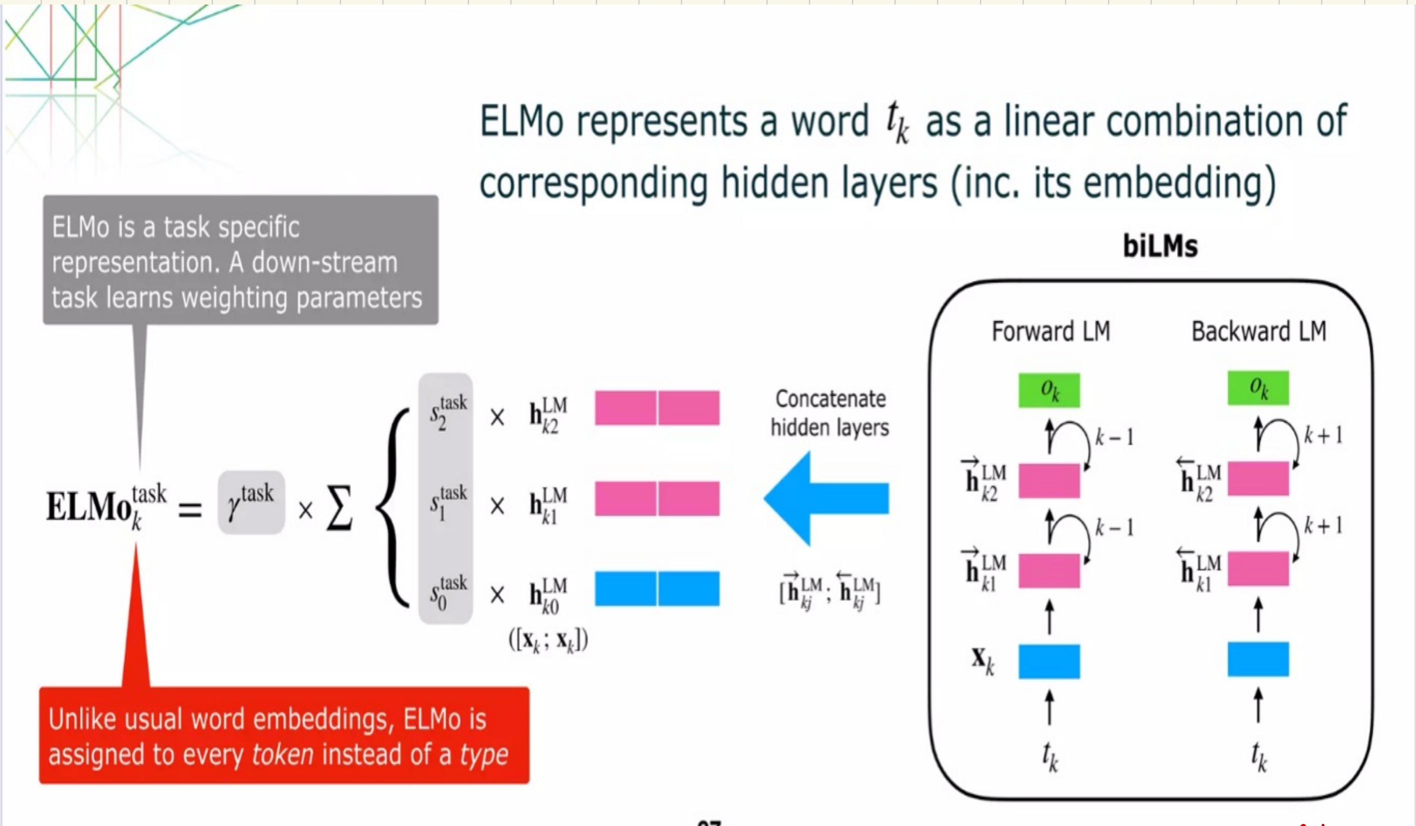


ELMo represents a word t_k as a linear combination of corresponding hidden layers (inc. its embedding)

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

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

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

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

biLMs
Concatenate hidden layers
$[\overrightarrow{\mathbf{h}}_{kj}^{LM} ; \overleftarrow{\mathbf{h}}_{kj}^{LM}]$
Forward LM
Backward LM
o_k
$k-1$
$k+1$
$\overrightarrow{\mathbf{h}}_{k2}^{LM}$
$\overrightarrow{\mathbf{h}}_{k1}^{LM}$
$\overleftarrow{\mathbf{h}}_{k2}^{LM}$
$\overleftarrow{\mathbf{h}}_{k1}^{LM}$
$\mathbf{x}_k$
t_k

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

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

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

$$h_{jk}^{LM} = [\overrightarrow{h}_{kj}^{LM}, \overleftarrow{h}_{kj}^{LM}]$$

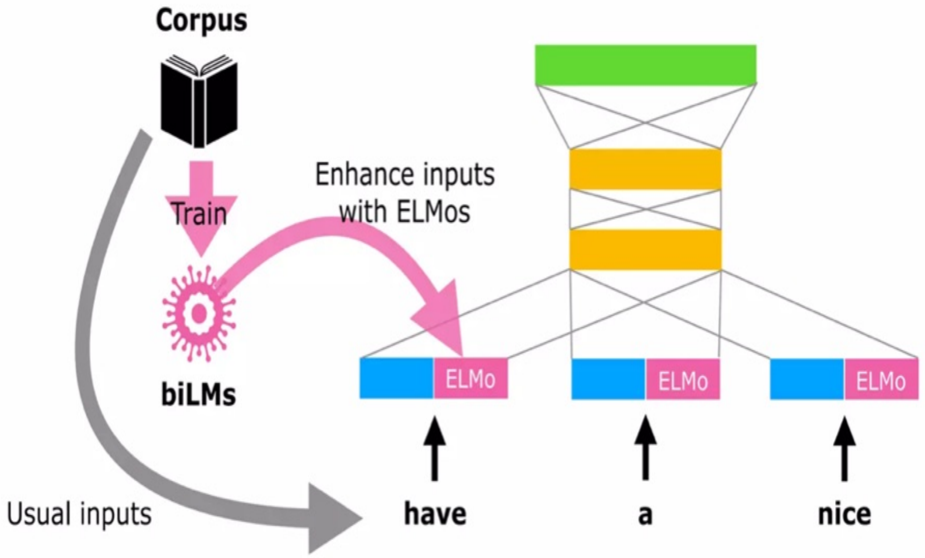
for each biLSTM layer.

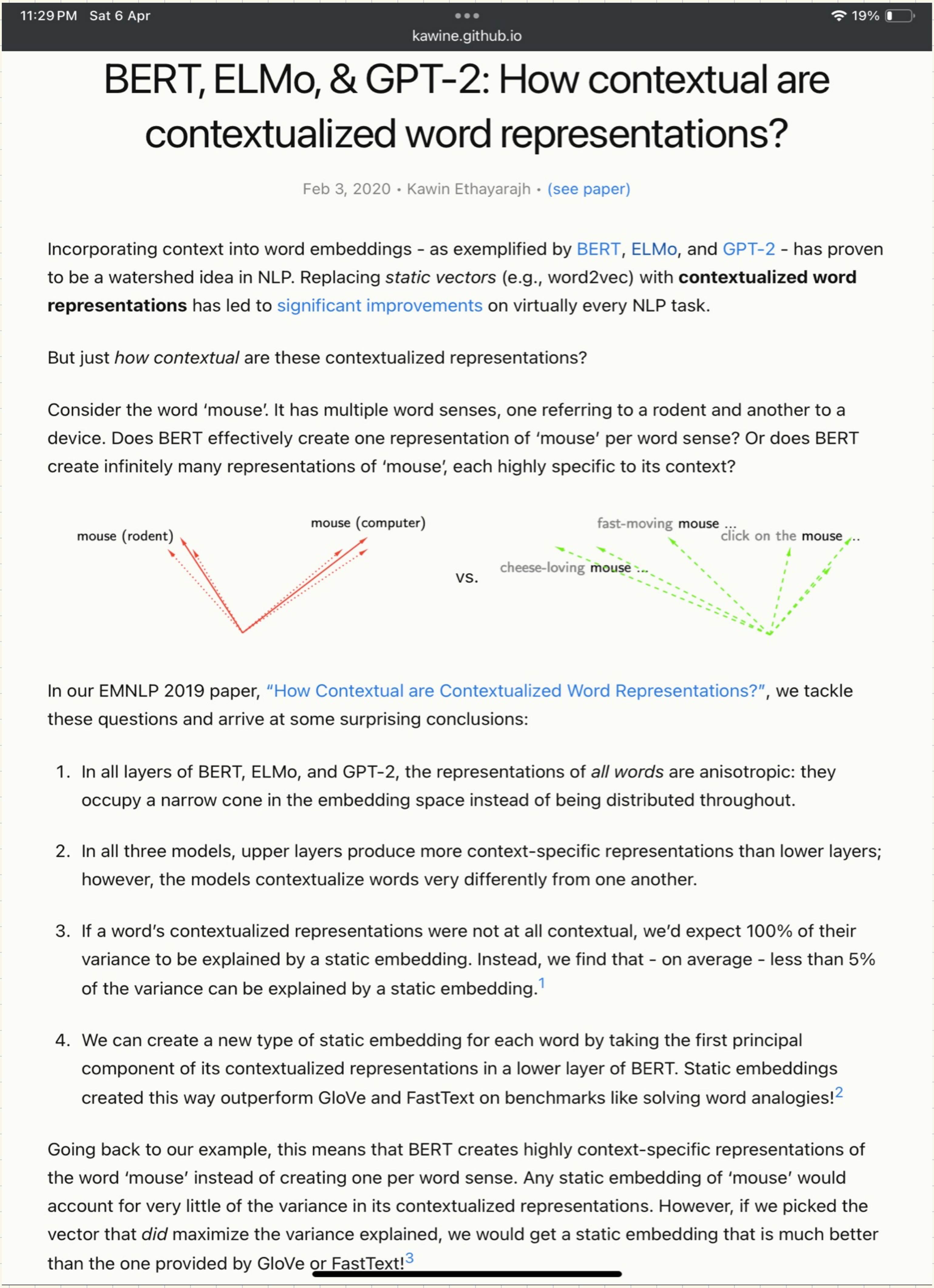
Method

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

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

- Overview
- Method
- Evaluation
- Analysis
- Comments





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

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

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

But just *how contextual* are these contextualized representations?

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

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

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

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

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

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

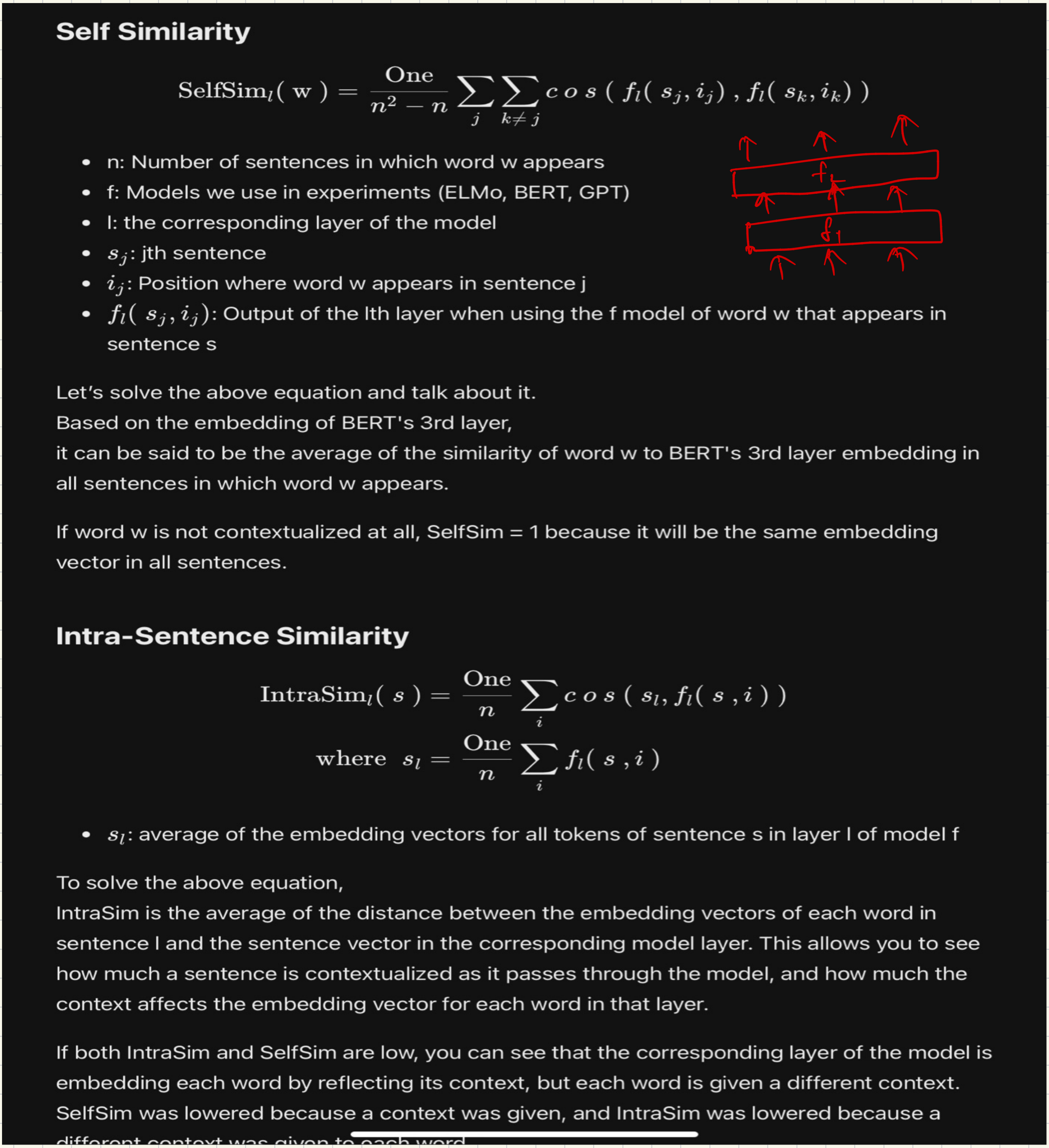
Going back to our example, this means that BERT creates highly context-specific representations of the word 'mouse' instead of creating one per word sense. Any static embedding of 'mouse' would account for very little of the variance in its contextualized representations. However, if we picked the vector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

2-3. Measures of Contextuality

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

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

Let's look at each one one by one.



Self Similarity

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

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

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

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

Intra-Sentence Similarity

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

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

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

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

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

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

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

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

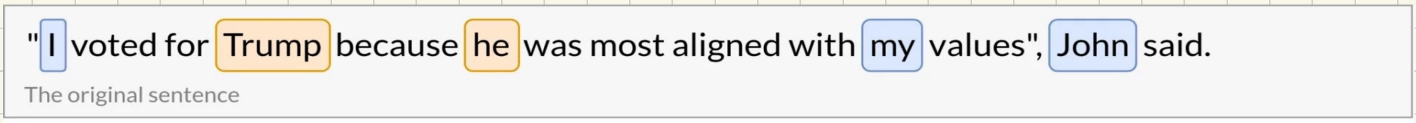
Contextualization = original meaning of the word + meaning of the entire sentence

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

BERT (By Google)

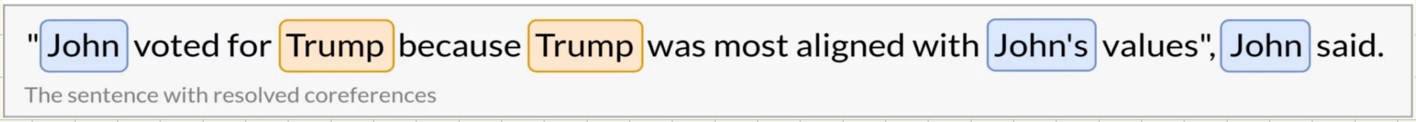
What is coreference resolution?

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



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

The original sentence



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

The sentence with resolved coreferences

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