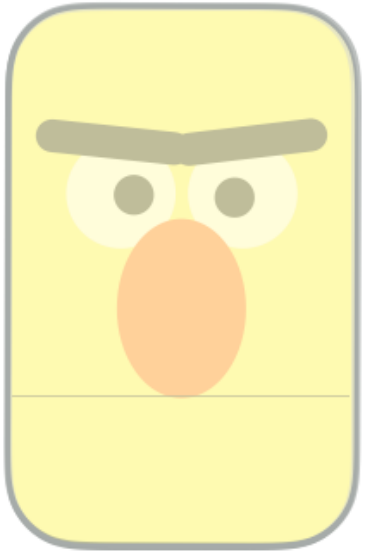
BERT
(Bidirectional Transformers for Language Understanding)

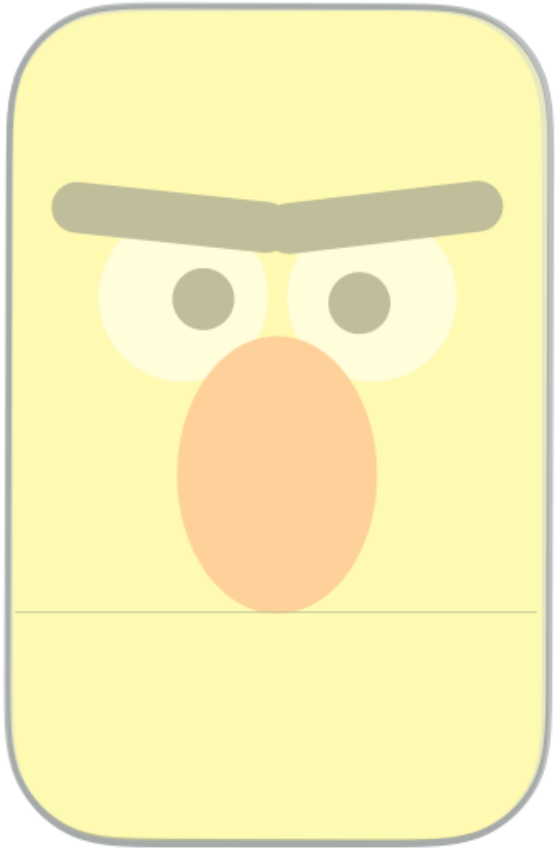
BERT trains language model in both directions

BERT trains Language Model in both directions

BERT

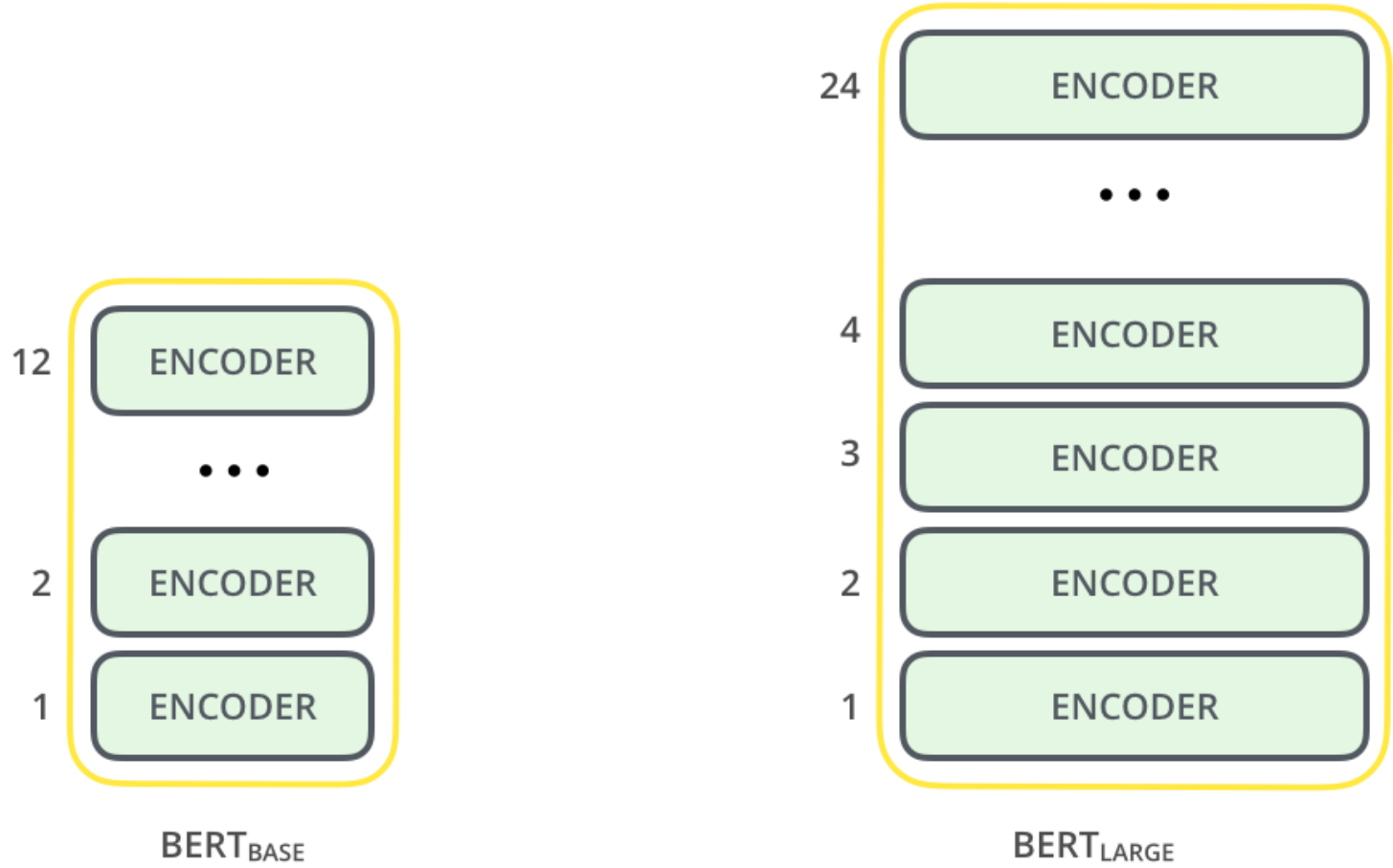


BERT$_{BASE}$



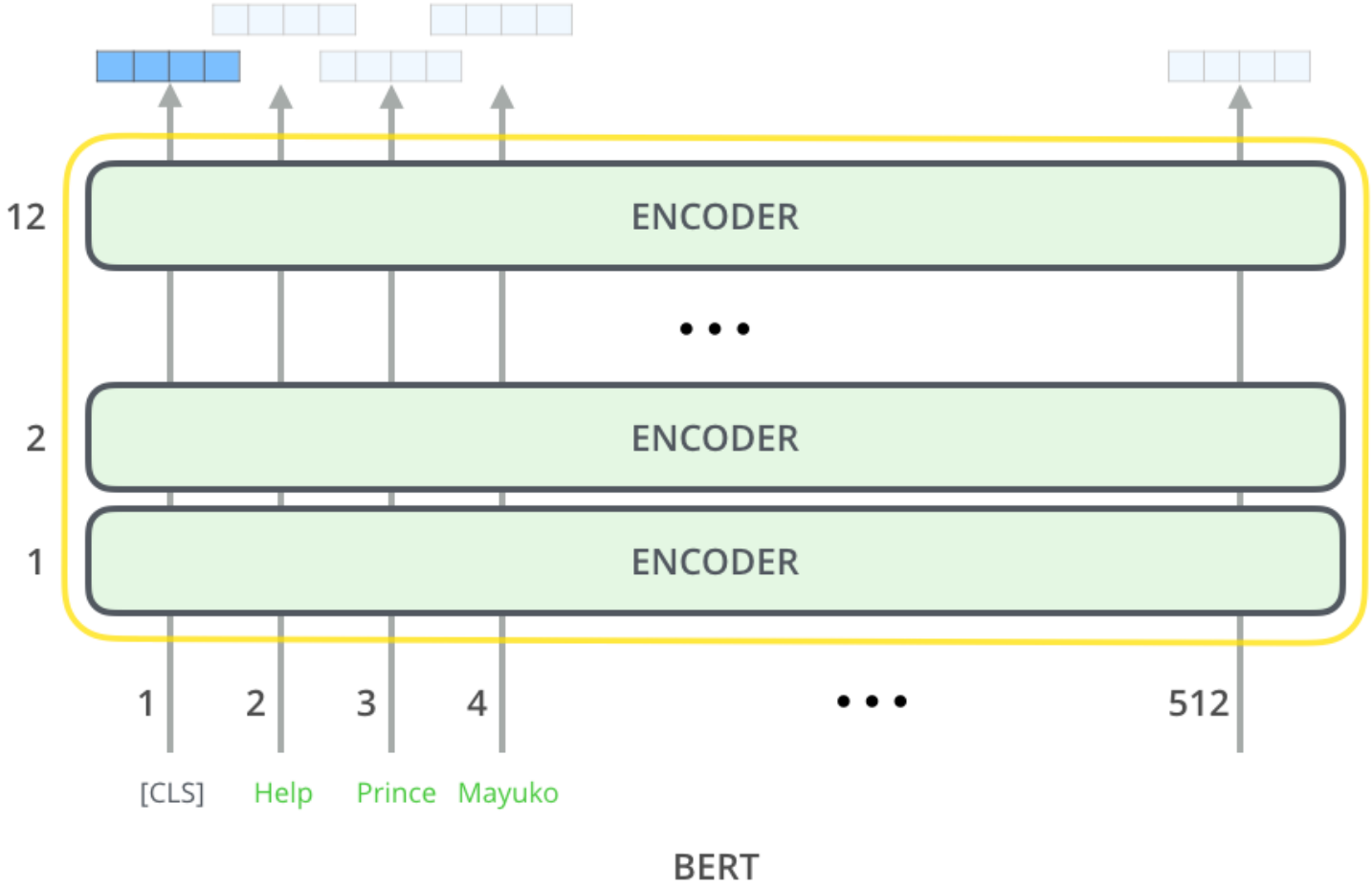
BERT$_{LARGE}$

BERT: Only Encoders are used



These also have larger feedforward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively)

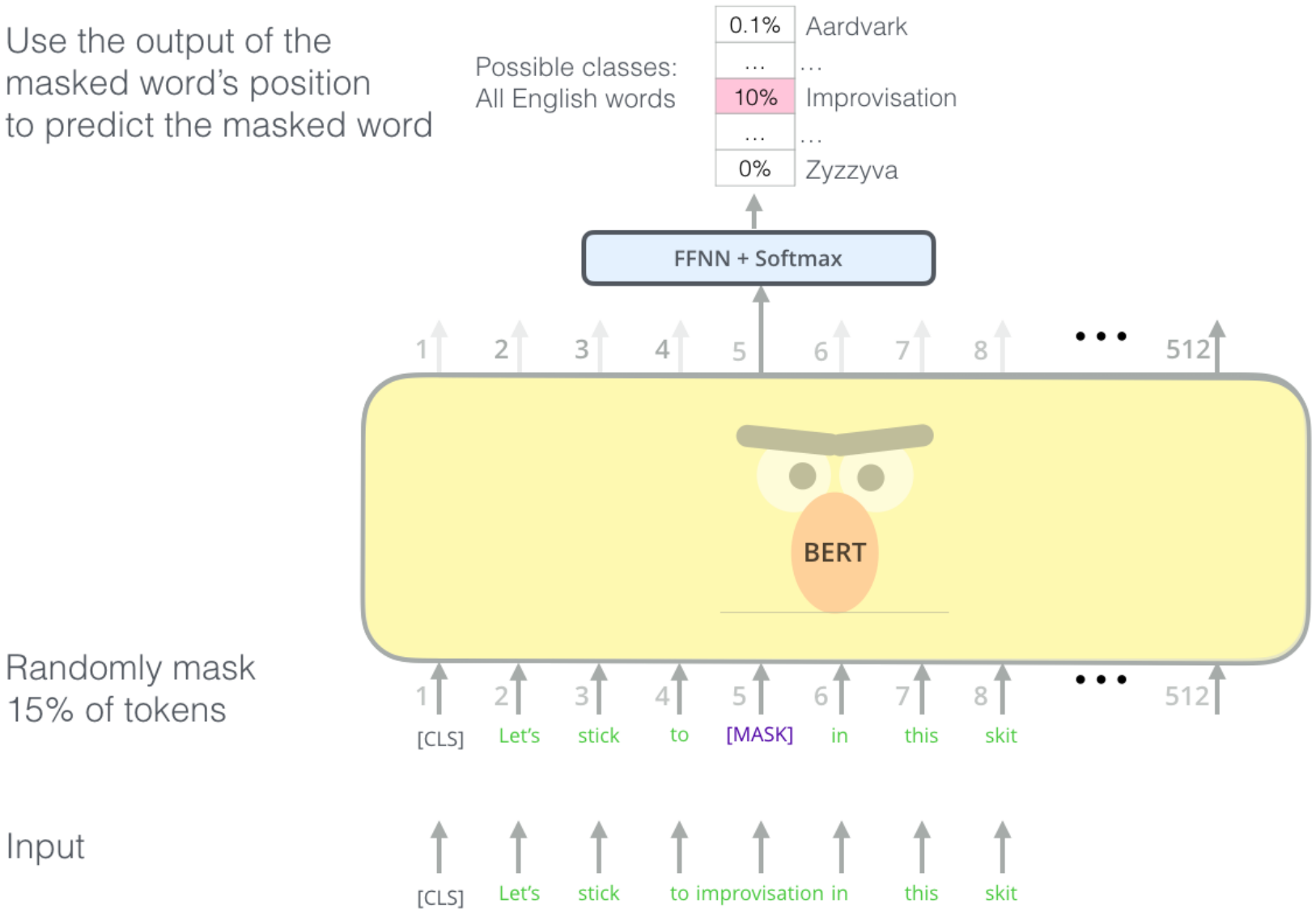
BERT: Only Encoders are used



BERT: For Language Modeling

- **Use [MASK] token for 15% for text**
- In this 15% tokens, 80% will be replaced with ‘[MASK]’, 10% will be replaced with a random word from vocabulary, 10% will not be replaced.

language modeling: Use [MASK] token for 15% for text



All 15% are not MASKED

- 80% were replaced by the ‘<MASK>’ token

Example: “My dog is ***<MASK>****”*

- 10% were replaced by a random token

Example: “My dog is **apple***”*

- 10% were left intact

Example: “My dog is **hairy***”*

Why did they not use a ‘<MASK>’ replacement token all around?

- If the model had been trained on only predicting ‘<MASK>’ tokens and then never saw this token during fine-tuning, it would have thought that there was no need to predict anything and this would have hampered performance
- By sometimes asking it to predict a word in a position that did not have a ‘<MASK>’ token, the model needed to learn a contextual representation of *all* the words in the input sentence, just in case it was asked to predict them afterwards.

- *Are not random tokens enough? Why did they leave some sentences intact?*

Ideally we want the model's representation of the masked token to be better than random. By sometimes keeping the sentence intact (while still asking the model to predict the chosen token) the authors biased the model to learn a meaningful representation of the masked tokens.

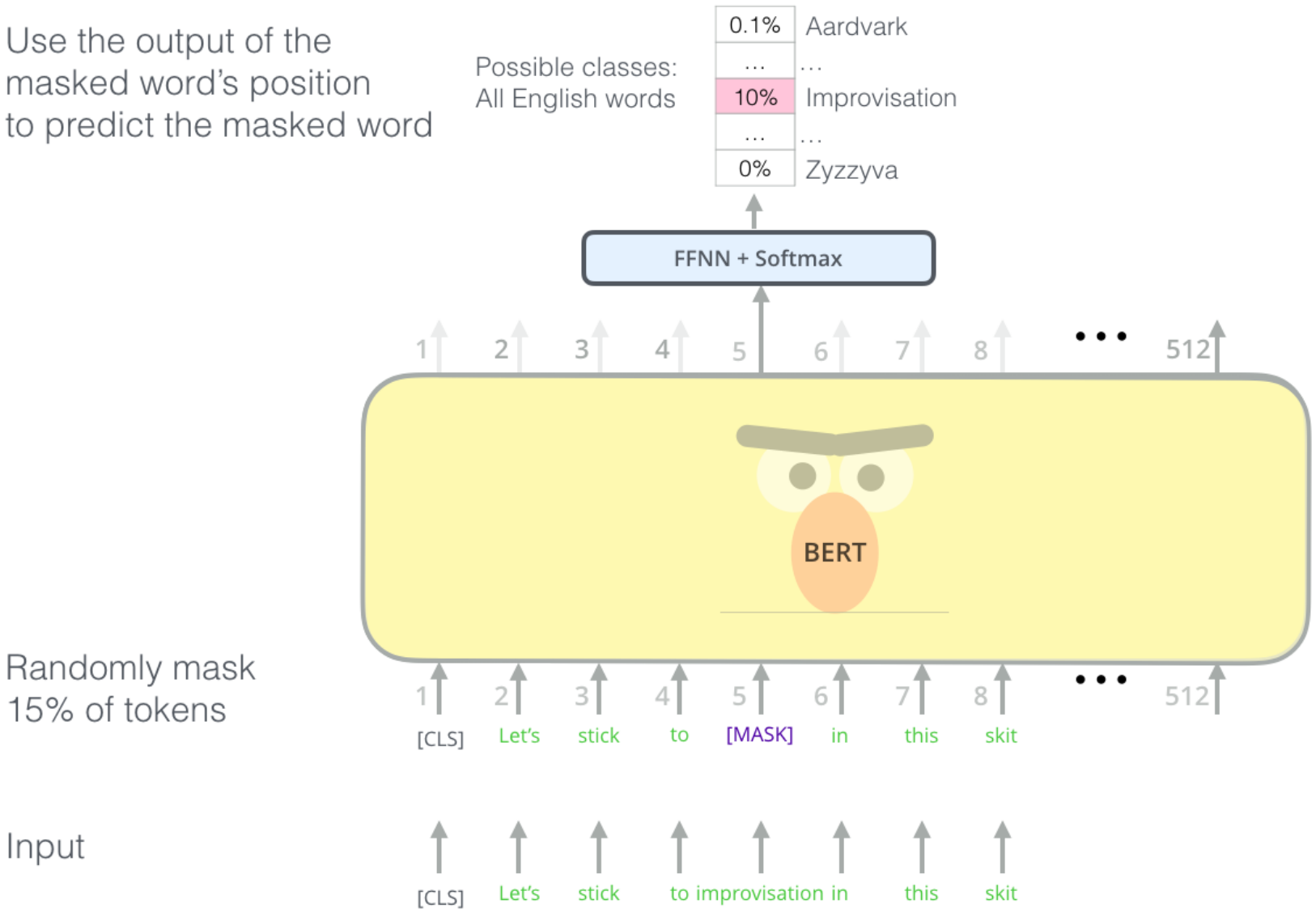
Will random tokens confuse the model?

The random replacement happened in 1.5% of the tokens (10%*15%) and the authors claim that it did not affect the model's performance.

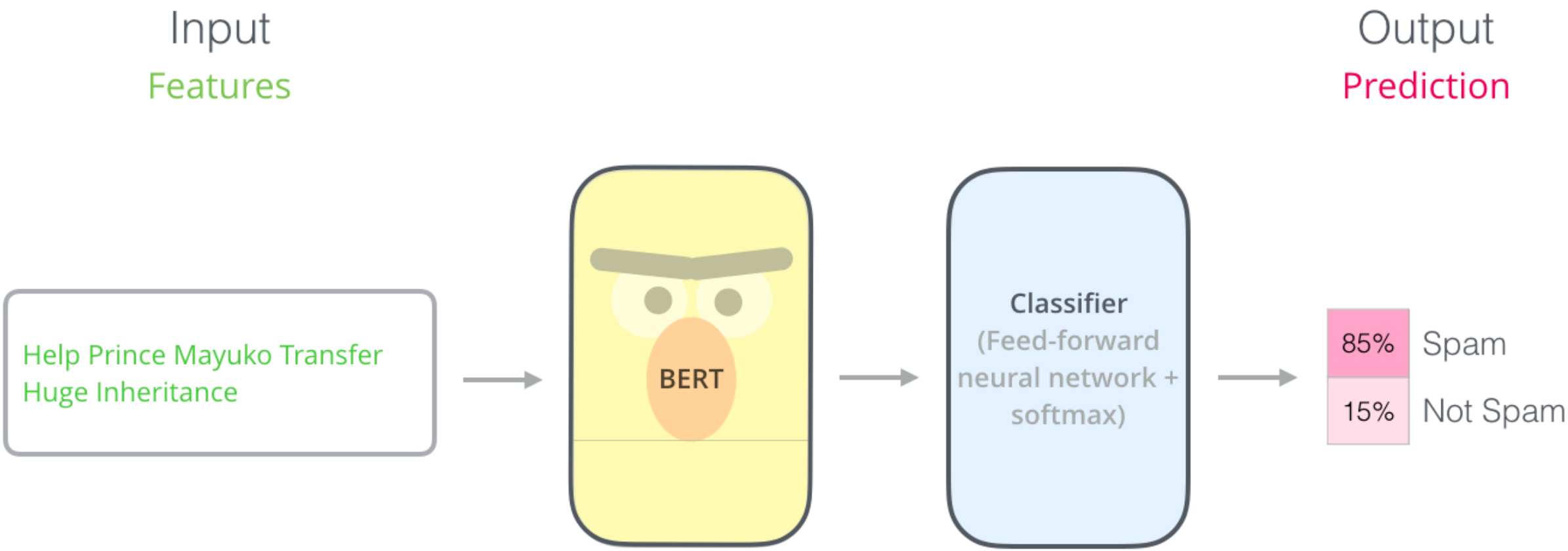
- *The model will only predict 15% of the tokens but language models predict 100% of tokens, does this mean that the model needs more iterations to achieve the same loss?*

- Training will be slower as model only predicts 15% of words

language modeling: Use [MASK] token for 15% for text



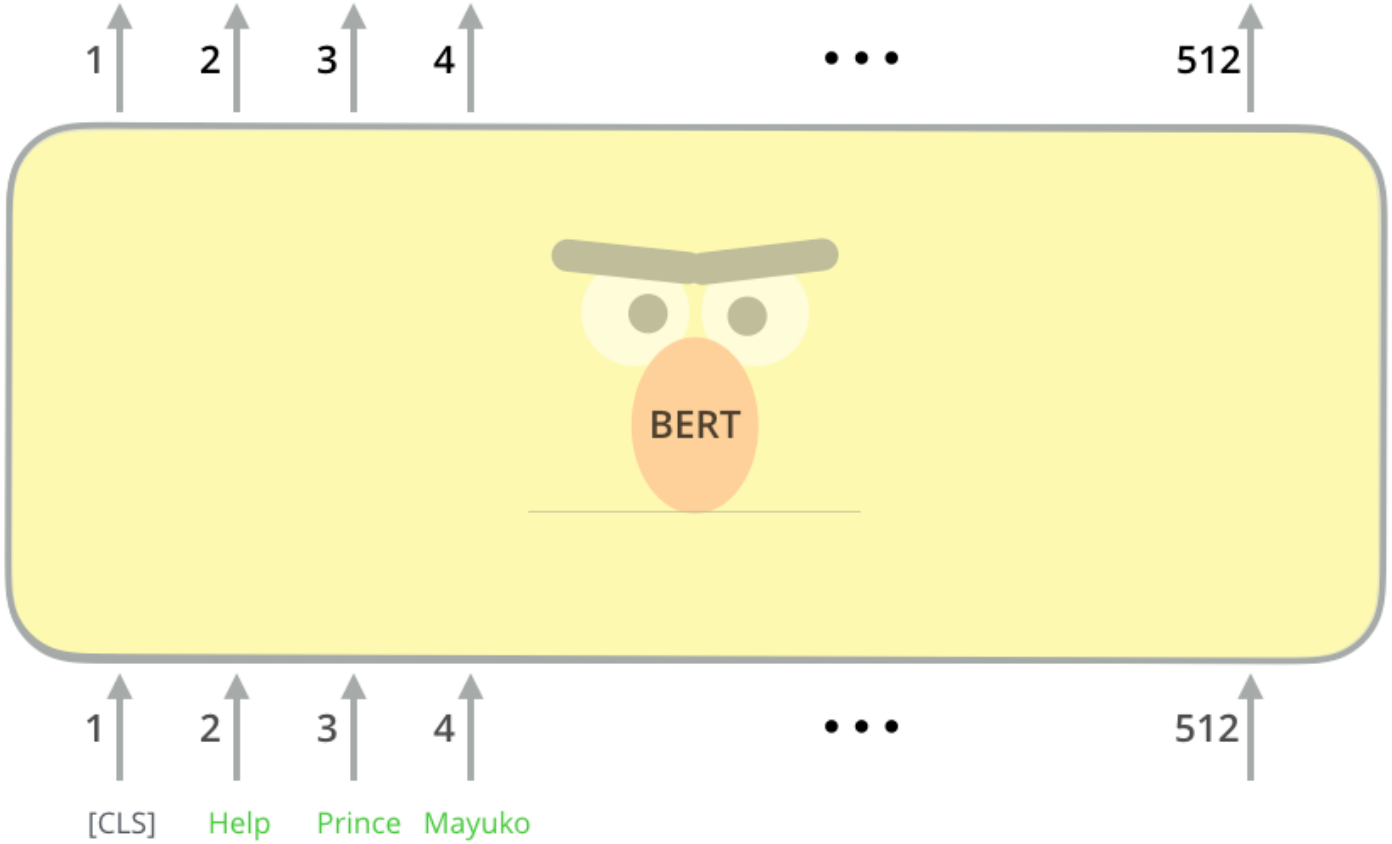
BERT: Sentence Classification



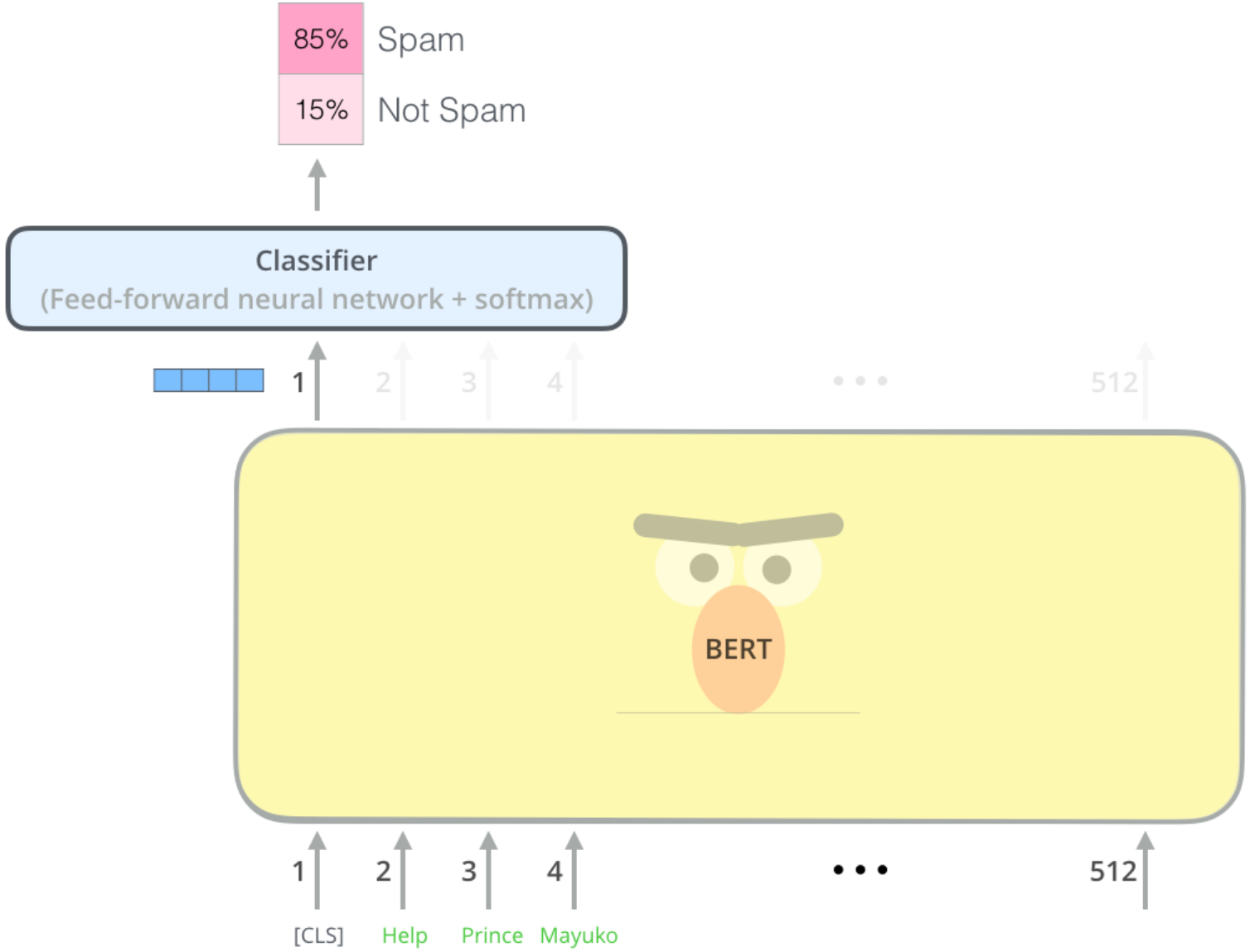
BERT: Sentence Classification

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

BERT: A special [CLS] token is used for classification



BERT: A special [CLS] token is used for classification



BERT: Two-sentence Tasks

- The sentence pair contains 2 sentences, 50% of the sentence pairs are related sentences which appears in the document one by one, 50% of the sentence pairs are not related, which the sentence are combined randomly.
- Useful for many NLP tasks like QA, entailment, inference etc

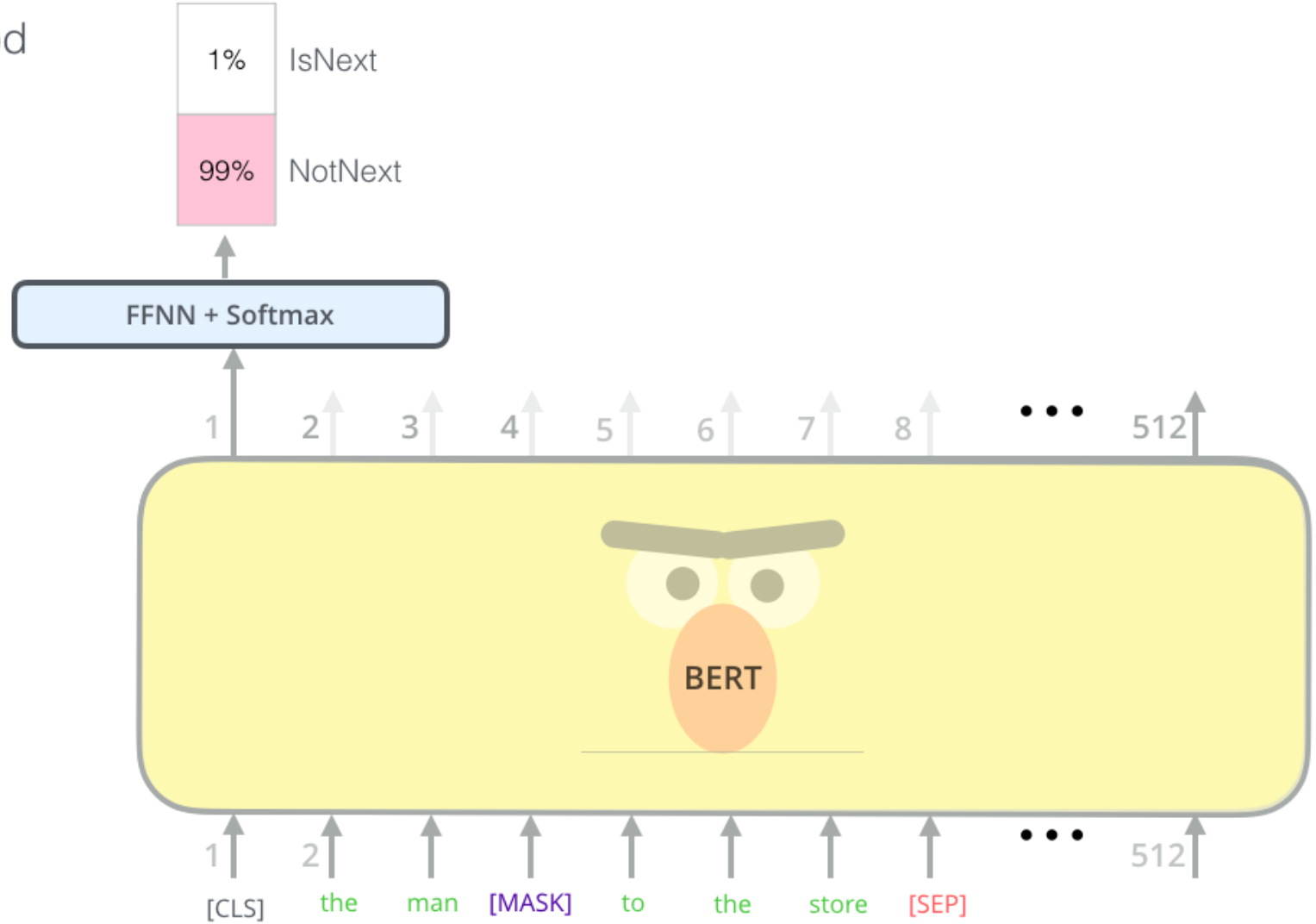
BERT: Two-sentence Tasks

- *Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]*
- In this case the sentences are adjacent, so the label in [CLS] would be ‘<IsNext>’ as in:
- *Input = <IsNext> the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]*

BERT: Two-sentence Tasks

- The loss was calculated as the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood.

Predict likelihood that sentence B belongs after sentence A



Tokenized Input

Input

[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]

Sentence A

Sentence B

Segmentation Embedding

- E.g. token sentence:
- ‘[CLS] I have a dream [SEP] The cat is white [SEP]’,
- segmentation embedding: [0,0,0,0,0,1,1,1,1,1]

Mask word embedding

- 0 represents normal word, 1 represents mask word.

Out of vocabulary words

- BERT Tokenization: WordPiece model

BERT Tokenization: WordPiece model

- This model greedily creates a fixed-size vocabulary of individual characters, subwords, and words that best fits our language data. Since the vocabulary limit size of BERT tokenizer model is 30,000, the WordPiece model generated a vocabulary that contains all English characters plus the ~30,000 most common words and subwords found in the English

BERT Tokenization: WordPiece model

- Whole words
- Subwords occuring at the front of a word or in isolation.
- Subwords not at the front of a word, which are preceded by ‘##’ to denote this case
- Individual characters
- The word “embeddings” is represented:

  [‘em’, ‘##bed’, ‘##ding’, ‘##s’]

BERT Tokenization: WordPiece model

- Rather than assigning out of vocabulary words to a catch-all token like ‘OOV’ or ‘UNK,’ words that are not in the vocabulary are decomposed into subword and character tokens that we can then generate embeddings for.

Context Based Similarity

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

First 5 vector values for each instance of "bank".

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])

Context Based Similarity

- Similarity of ‘bank’ as in ‘bank robber’ to ‘bank’ as in ‘river bank’
  - Cosine similarity = 0.67

- Similarity of ‘bank’ as in ‘bank robber’ to ‘bank’ as in ‘bank vault’
  - Cosine similarity = 0.9

How to use pretrained model

- Fine Tune
- Feature based

BERT as Fine Tuned Model

- Add a classifier (like softmax layer) after the pretrianed model, then use downstream task data to learn new layer's weights and slightly change the weights of pretrained model with back propagation
- **Pros:** when we don't have enough labeled data for training, with transfer learning by pretrained model ,we can somehow increase our training model accuracy with little training data.
- **Cons:** Task's input embedding must be the same as the pretrained task, also, the task between downstream and pretrained task can not be too different, otherwise, what model have learned form pretrained task cannot apply to new task.

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:

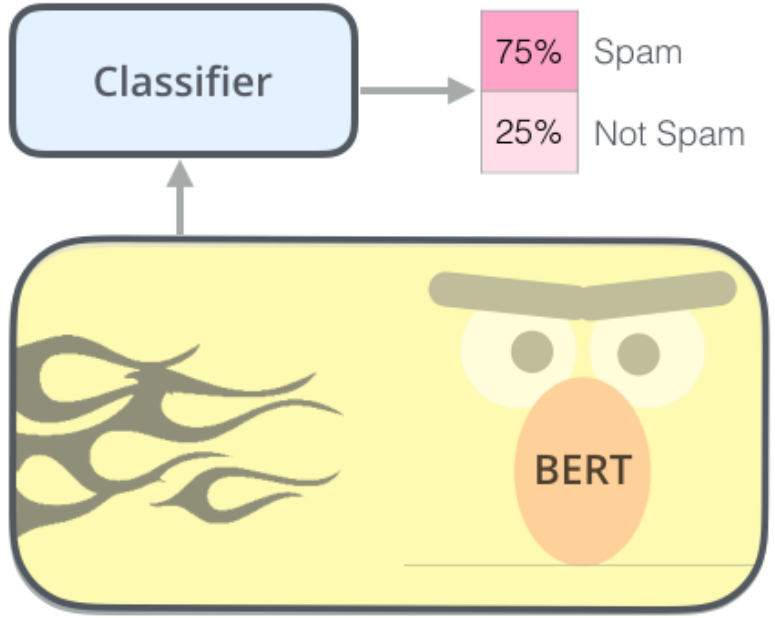
Objective:

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)



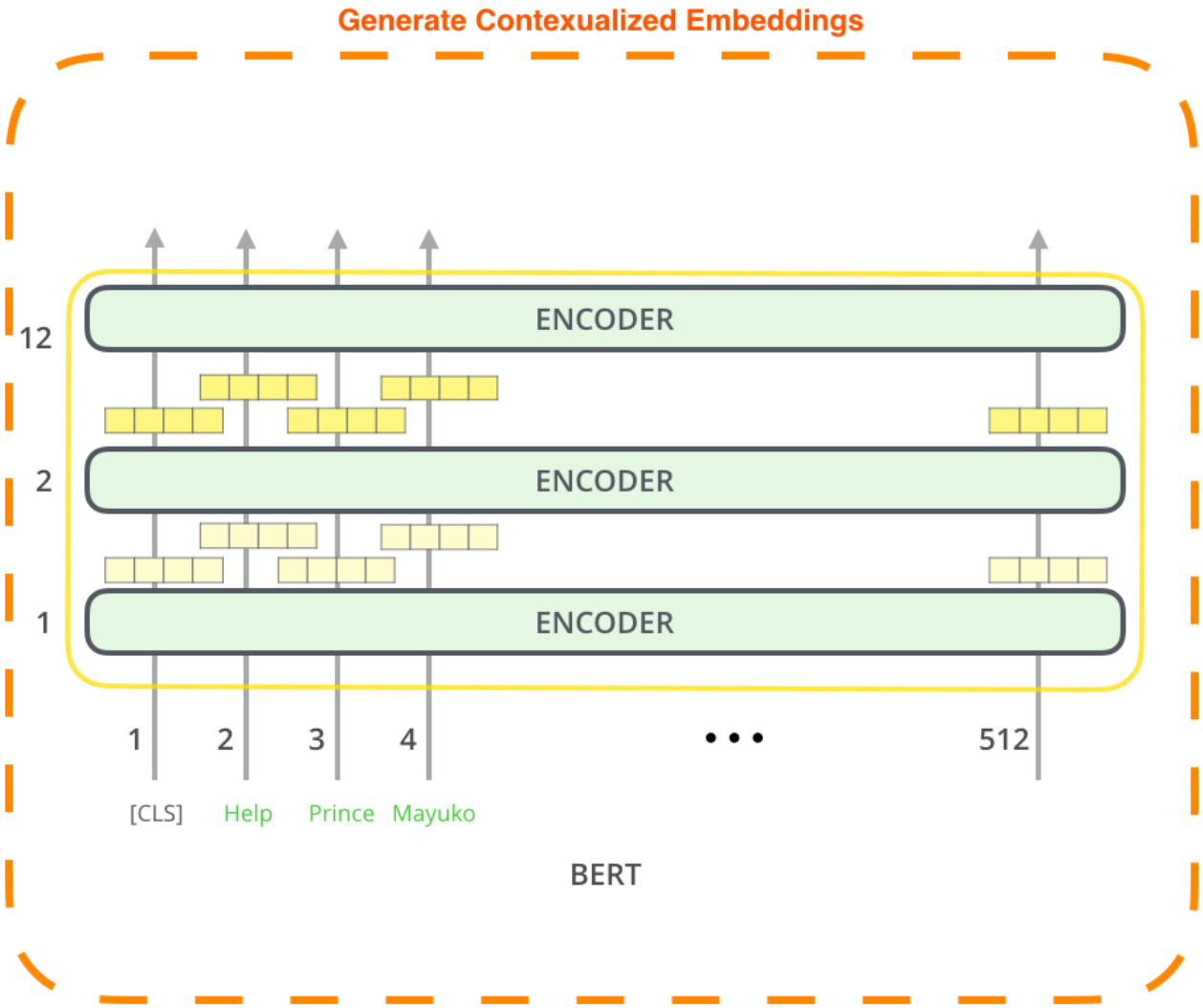
Dataset:

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

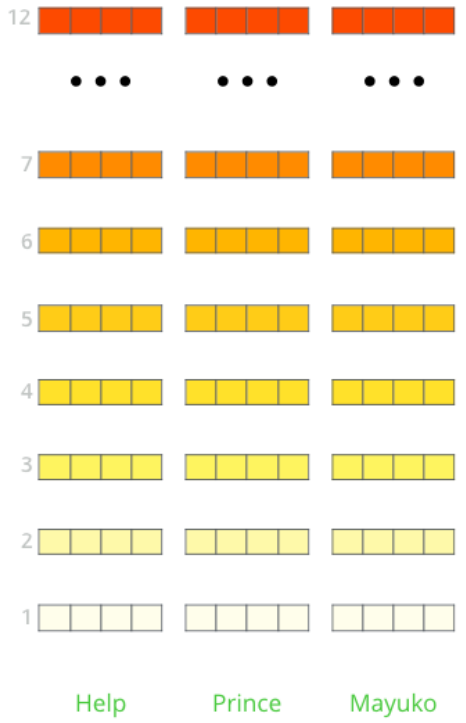
BERT as Feature Extraction Model

- Just like what W2V, we treat the pretrained result as a feature embedding for our downstream task, then we will use our full defined model(like: LSTM) to learn some abstract mapping knowledge from the embedding .
- **Pros:** when we get enough learning data, its ok to learn from pretrained embedding, we have little restrict from pretrained model ,since it just act like an input for the whole model
- **Cons:** we need to learn the model from scratch, and our model can be only used for only one task

BERT: For Feature Extraction



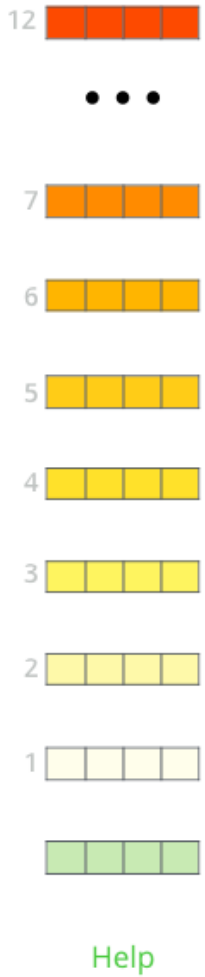
The output of each encoder layer along each token's path can be used as a feature representing that token.

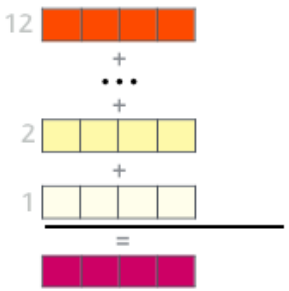
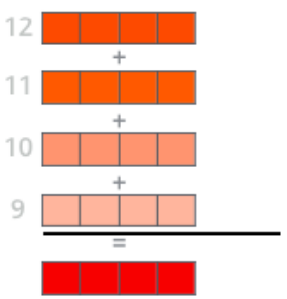
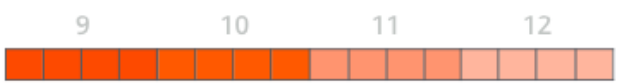


But which one should we use?

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

For named-entity recognition task CoNLL-2003 NER



		Dev F1 Score
First Layer		91.0
Last Hidden Layer		94.9
Sum All 12 Layers		95.5
Second-to-Last Hidden Layer		95.6
Sum Last Four Hidden		95.9
Concat Last Four Hidden		96.1

BERT: Multihead Redundancy

- The following paper has description of the regularization term added to the loss for ensuring that different attention heads learn different distributions. The regularization term makes sure attention distributions are different and focus on only few items.
- A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING (https://arxiv.org/pdf/1703.03130.pdf)

- The following paper has visualization examples of different BERT attention heads paying attention to different words.
- What Does BERT Look At?An Analysis of BERT's Attention (https://nlp.stanford.edu/pubs/clark2019what.pdf)

References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*
- http://jalammar.github.io/illustrated-bert/
- https://medium.com/dissecting-bert/dissecting-bert-part2-335ff2ed9c73
- https://billpku.github.io/2019/04/20/BERT%20in%20theory/#About-BERT-masked-LM-task
- https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/