Economics 2355: Understanding Transformers, Visualization, and Sentiment Analysis

Melissa Dell

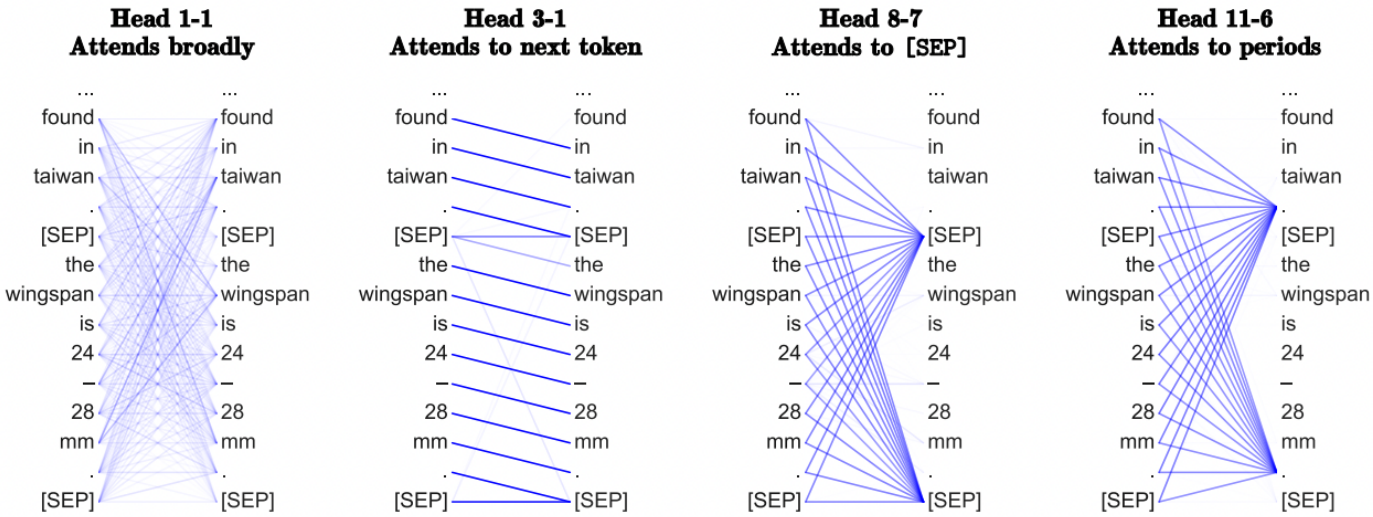
Harvard University

March 2021

What does BERT attend to?

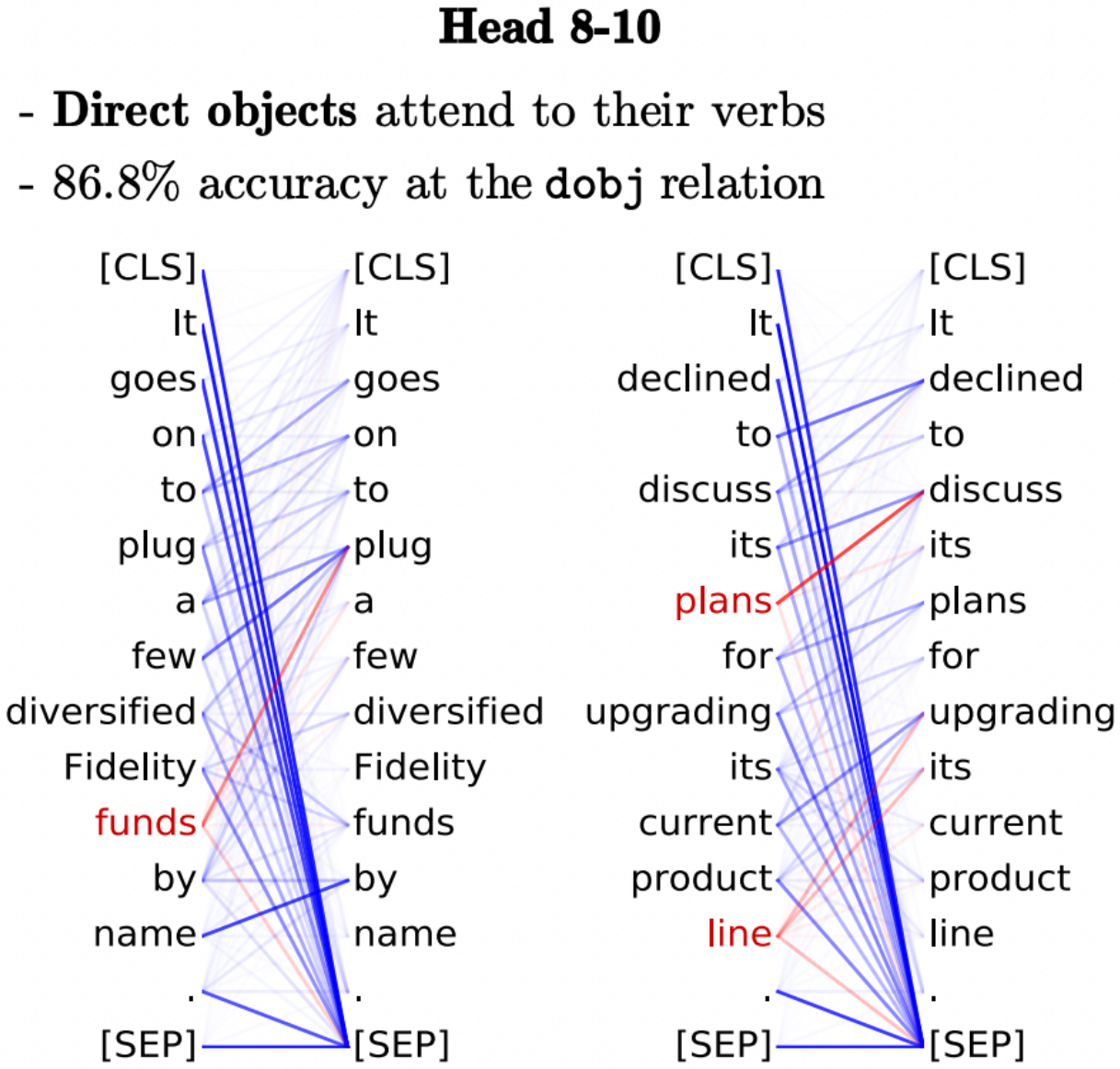
- There is a literature that seeks to understand the inner workings of Transformer-based language models
- Clark, Khandelwal, Levy, and Manning examine what pre-trained BERT attends to. They are *not* looking at a version of BERT that's been fine-tuned on any downstream task. That is, the model has only seen self-supervised training
- **Bottom line:** BERT's attention heads attend to linguistic phenomena very well, despite not being trained on them

What does BERT attend to?



Clark et. al

What does BERT attend to?

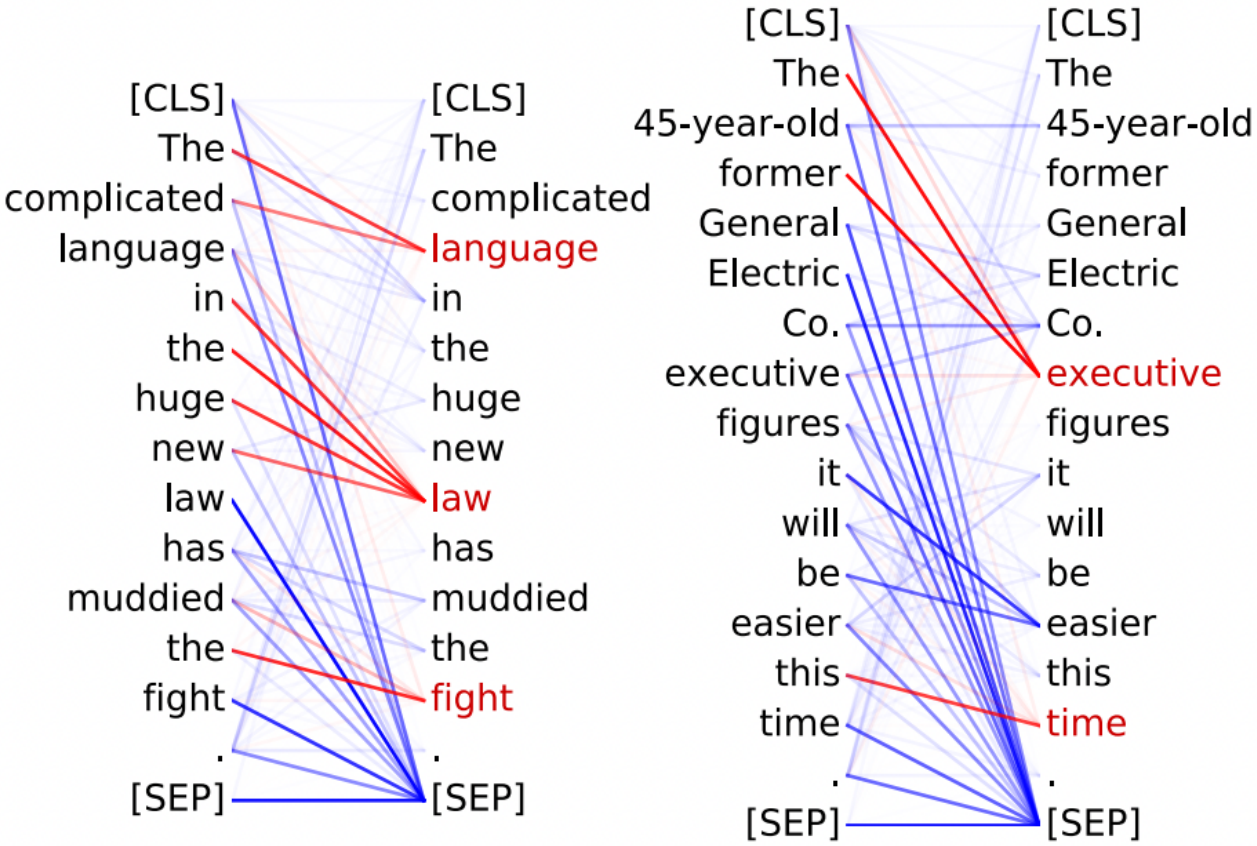


Clark et. al

What does BERT attend to?

Head 8-11

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the **det** relation

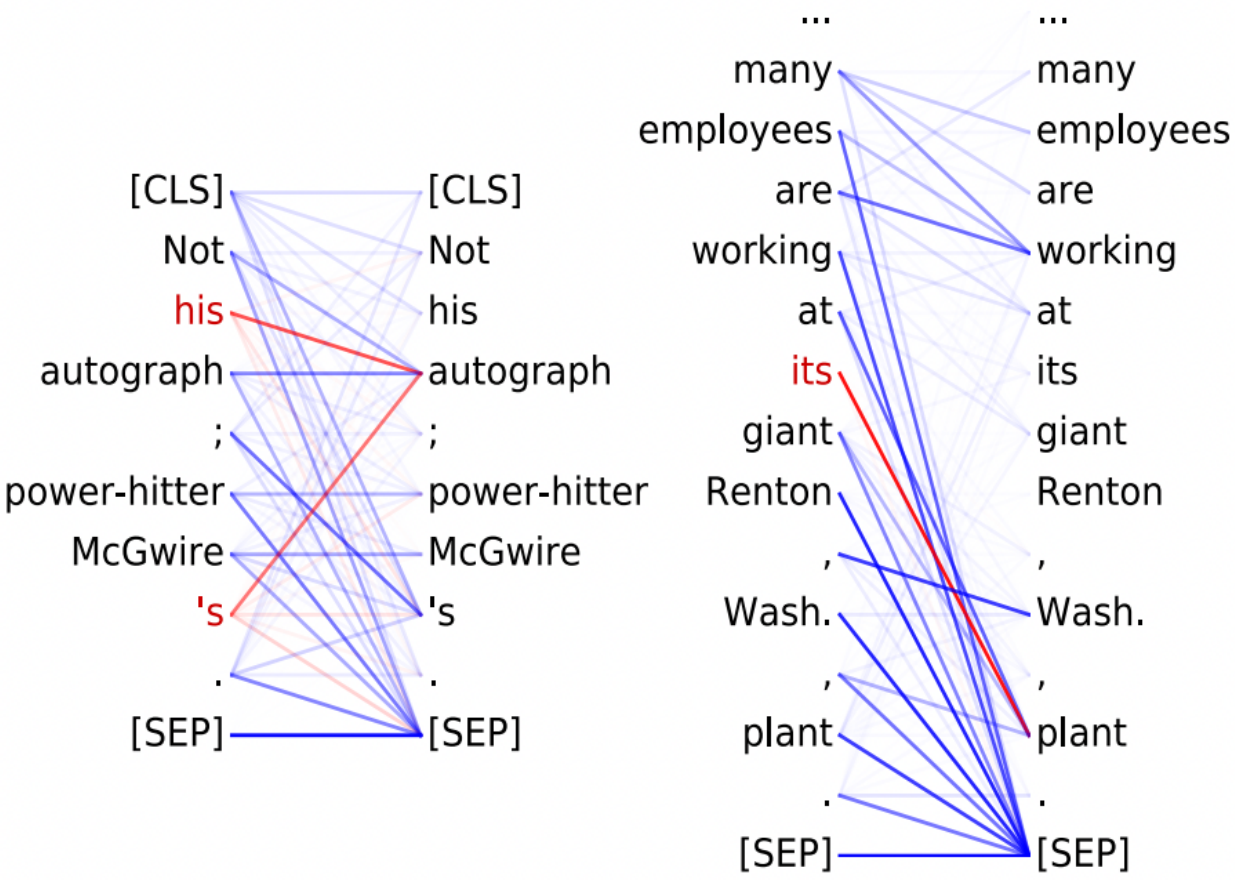


Clark et. al

What does BERT attend to?

Head 7-6

- **Possessive pronouns** and apostrophes attend to the head of the corresponding NP
- 80.5% accuracy at the **poss** relation

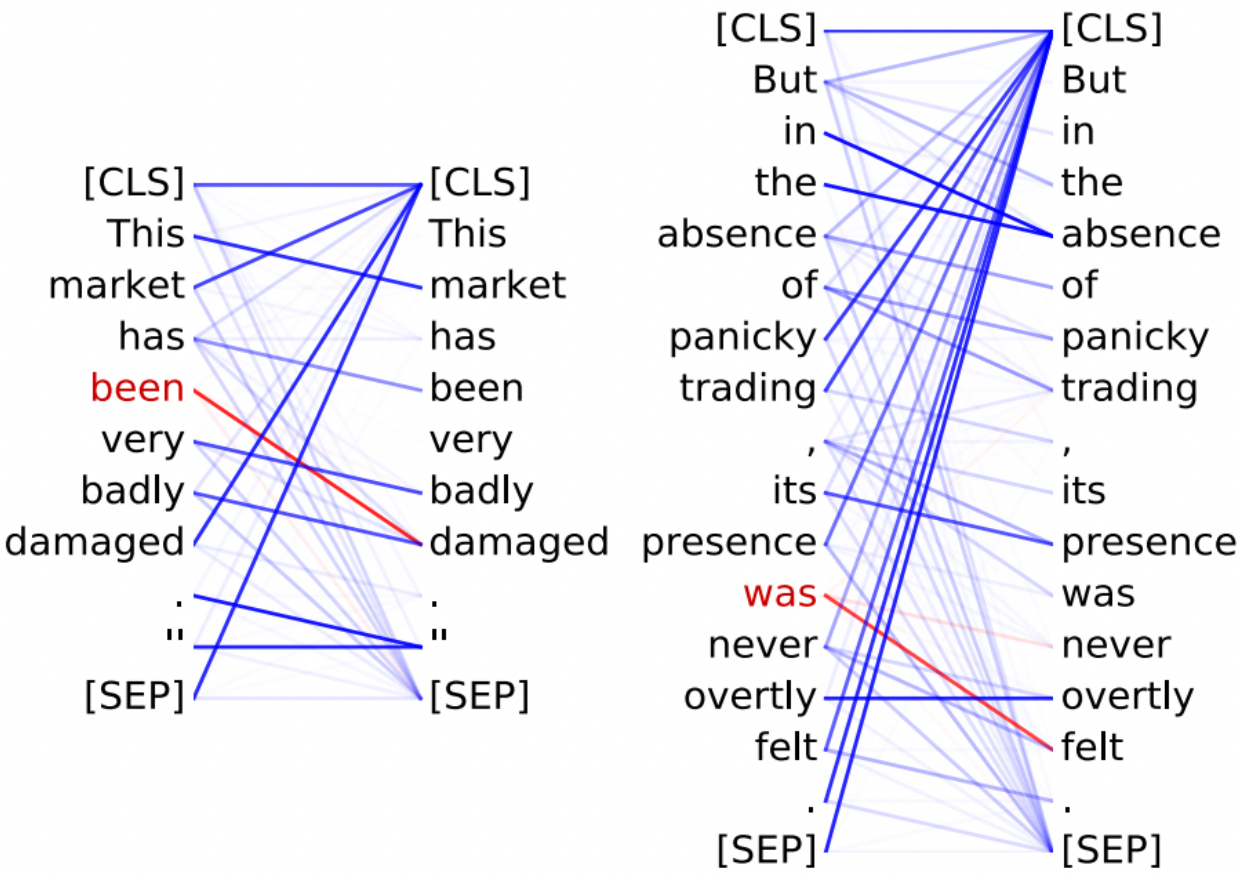


Clark et. al

What does BERT attend to?

Head 4-10

- **Passive auxiliary verbs** attend to the verb they modify
- 82.5% accuracy at the **auxpass** relation

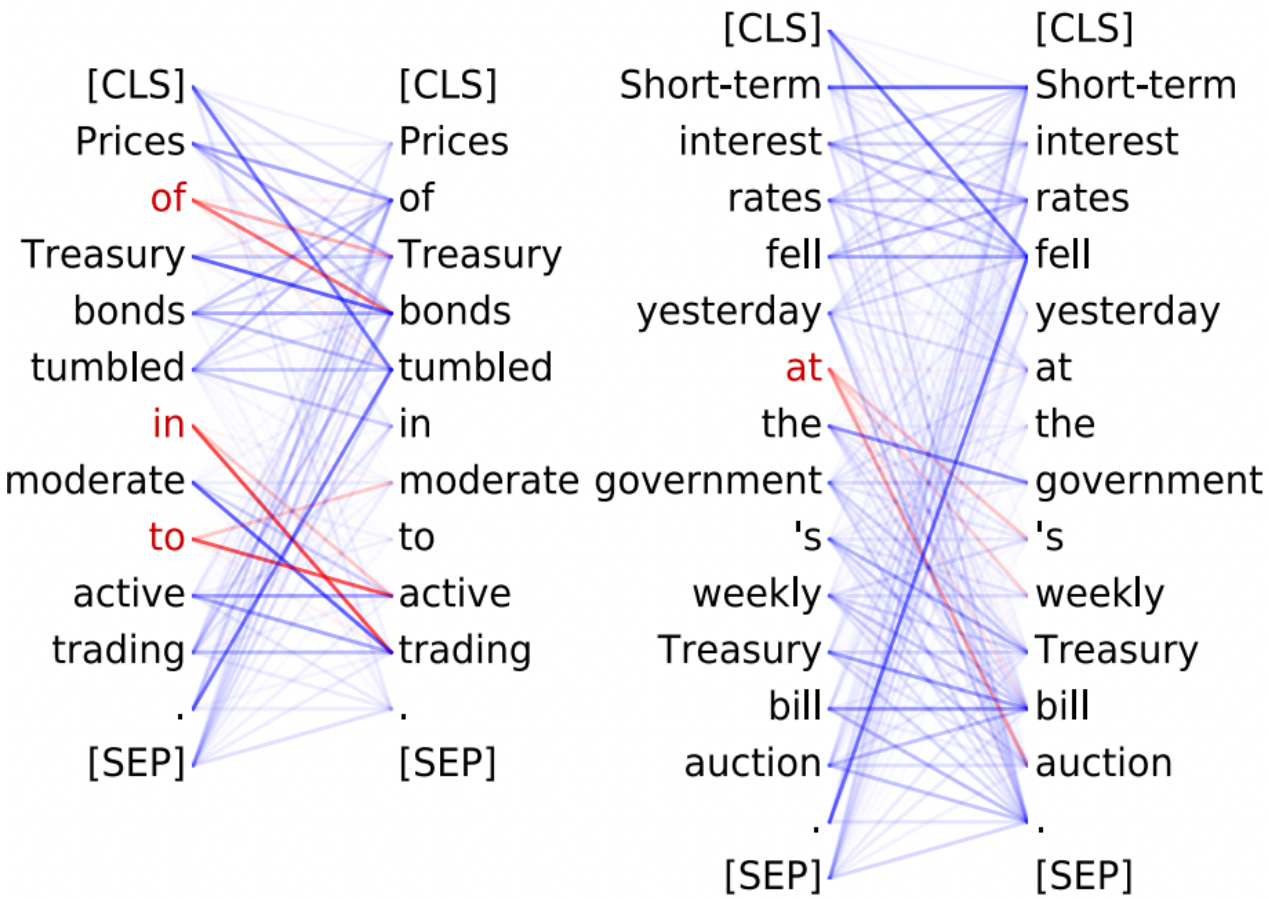


Clark et. al

What does BERT attend to?

Head 9-6

- **Prepositions** attend to their objects
- 76.3% accuracy at the **pobj** relation



Clark et. al

Outline

What's in an Embedding?

- Traditionally, people have tried to elucidate what's in an embedding by using probing classifiers, that try to predict certain things about words using just their pre-trained embeddings (i.e. PoS, tense, gender, word senses based on context, etc)
- A recent survey paper by Rogers et al. (2020) aggregates the findings of many different probing classifier studies of BERT
- The bottom line is that BERT embeddings contain information on a lot of stuff. If BERT embeddings can produce a coherent classifier, they provide information on that feature

Different Hidden Representations

- You can also probe what's contained in hidden representations from different layers of BERT
- There's some evidence that sentiment evaluation peaks with the later layers, though this appears to not have been very thoroughly studied
- More generally, representations of the same word in different contexts move apart deeper into the network, suggesting the upper layers produce more context-specific representations

What Happens to BERT embeddings during fine-tuning?

- A paper on this topic by Merchant et al. suggests that fine-tuning mostly alters the top few layers
- Fine-tuning has a significant effect on in domain sentences, but out-of-domain sentences remain much closer to the original representations

Outline

Distillation of Embeddings

Embeddings are dense objects in high-dimensional space (i.e. 768 dimensions for BERT-base and 1024 dimensions for BERT-large). They require processing to allow useful analysis. There are two main approaches to distilling relevant information from an embedding:

- Dimensionality reduction
- Distillation via function: use a classifier or some other function to extract relevant information (the approach to sentiment analysis)

Dimensionality Reduction: PCA

Three main approaches:

- PCA (don't use)
- tSNE
- UMAP (SoTA)

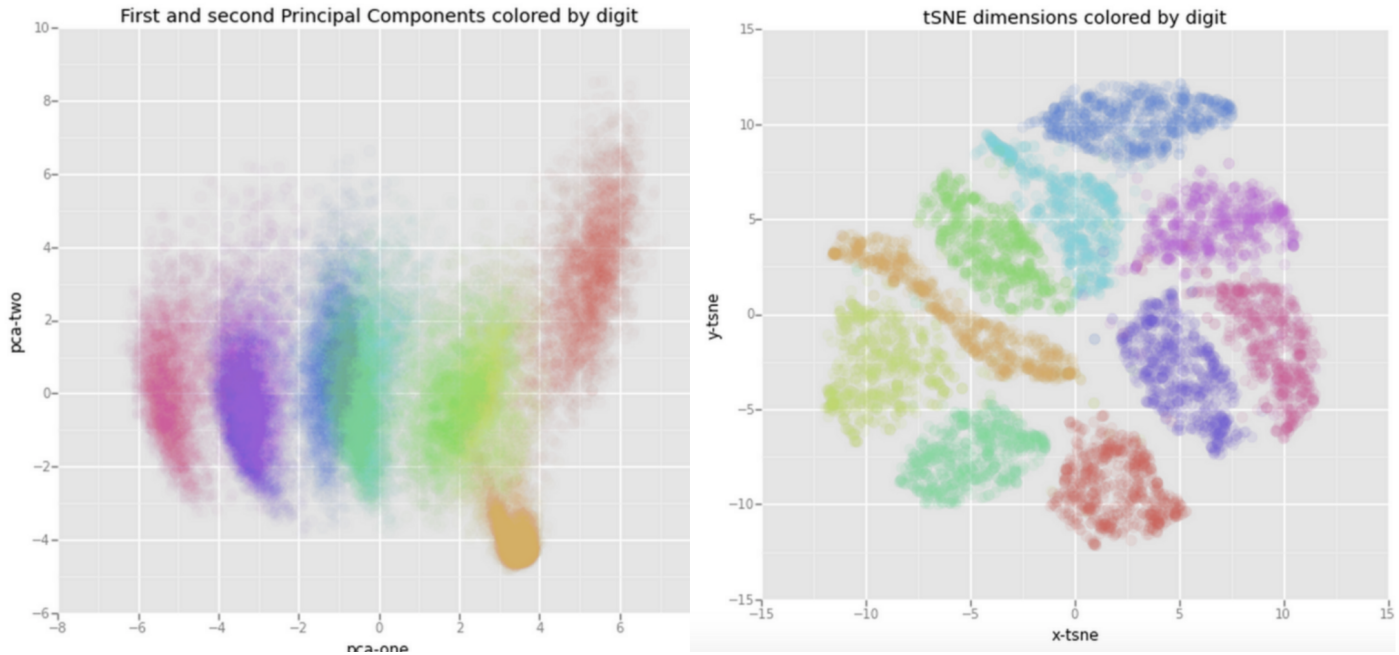
Dimensionality Reduction: PCA

- Finds the orthogonal bases (axes) that maximize the variance of the data and and then projects the data onto those axes
- The first principal component is the direction of greatest variance in the data
- Not optimized for preserving the local or global structure of a set of high dimensional data points

Dimensionality Reduction: tSNE

- tSNE stands for t-Distributed Stochastic Neighbor Embedding
- Creates probability distributions over neighboring points in a high dimensional space (for which high probabilities correspond to closer points in the space) and then tries to recreate the distributions for these points in a much lower (2-3) dimensional space
- Good at preserving local structures but not global structures - i.e. it can separate out distinct clusters in the data but distances between them aren't very meaningful

What does BERT attend to?



Dimensionality Reduction: UMAP

- SoTA approach Fairly involved, but roughly speaking, the method creates a manifold in high dimensional space with good properties, creates a similar manifold in low dimensional space, and then maximizes their similarity using cross-entropy loss
- Preserves local structure well, like tSNE, and does better with global structure

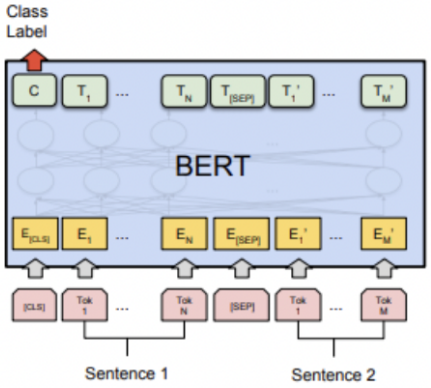
Probing Classifiers

- Probing classifiers also come into play for visualization
- Hewitt and Manning Methodology: estimate a matrix that can push BERT embeddings into a low-dimensional space where the distance between the projected embeddings corresponds to the distance between words in a syntax tree
- Can do this with other objectives - i.e. word sense disambiguation - and change the size of the matrices to successively squeeze the embeddings into lower and lower dimensions

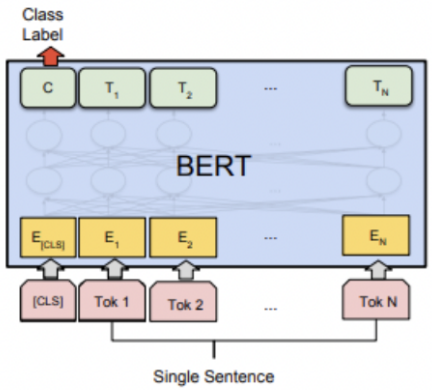
Outline

Sentiment

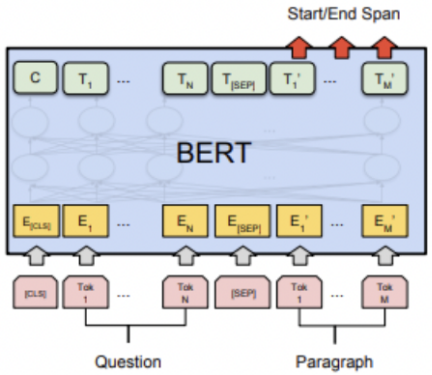
Modern language modeling makes it very straightforward to conduct sentiment analysis. For example, in BERT just fine tune the classifier on the CLS token.



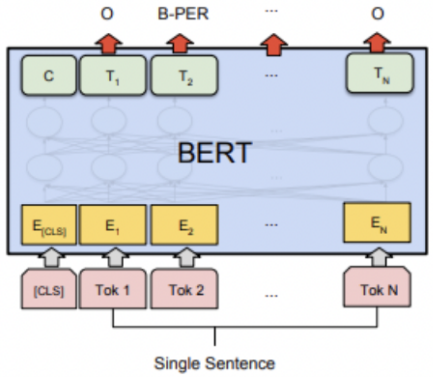
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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

Sentiment

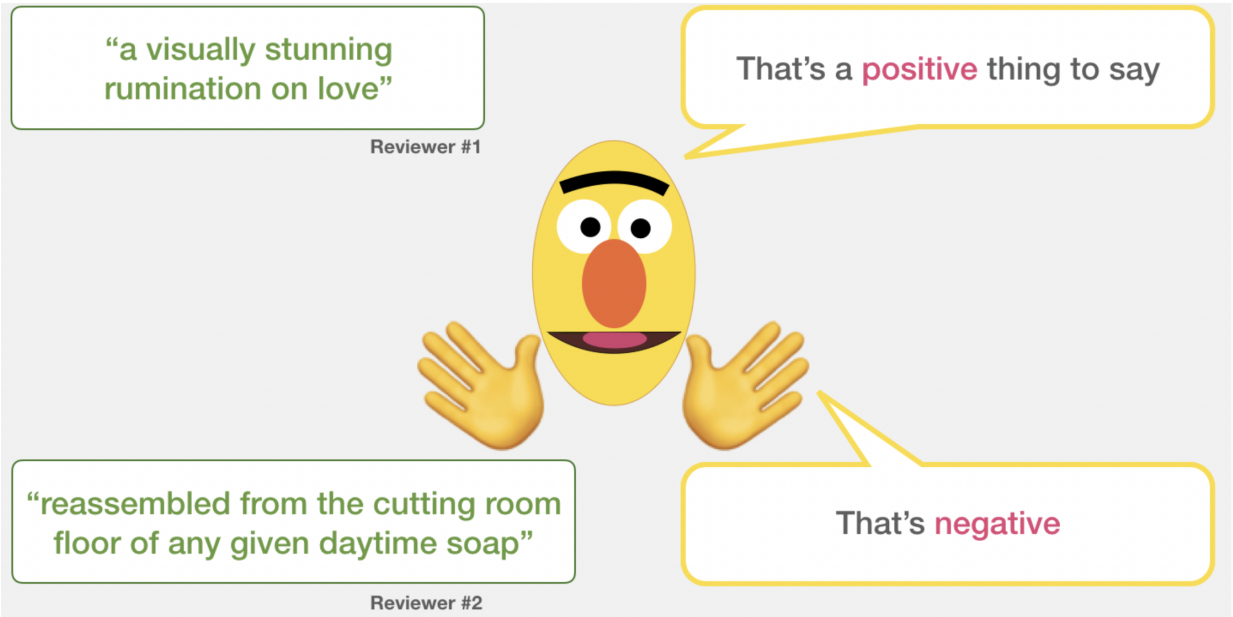
- The biggest challenge will likely be assembling labeled data to train the model
- We'll discuss zero-shot classification later in the course, but odds are you will still need some sort of labeled data to achieve desired accuracy
- One middle-of-the-road approach that can work well is to fine-tune first on a large, somewhat related/noiosy pre-existing/automated dataset and then fine-tune again on a smaller carefully curated dataset that relates closely to the task at hand

Sentiment Analysis Using the Embeddings

- Instead of fine-tuning, you could use the pre-trained embeddings from a LM to train a linear classifier (i.e. a FC layer followed by a softmax)
- This probably won't lead to quite as accurate performance and raises the question of which embeddings to choose
- However, it has the advantage of being very computationally lightweight
- Jay Alammar, who made the well-known Illustrated Transformer and Illustrated BERT pages, has a great illustration of how to do this (with code)

Sentiment Analysis Using Pre-trained Embeddings

The classic benchmark task for sentiment analysis is classifying movie reviews as positive or negative (i.e. SST2). This is the example considered here:



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

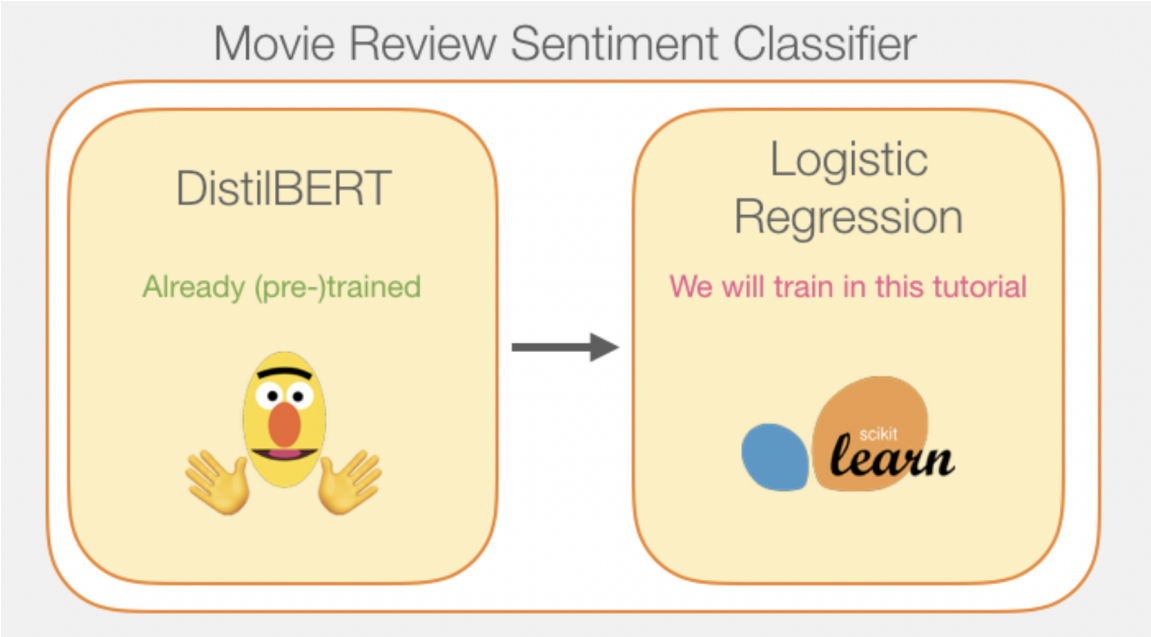
Sentiment Analysis Benchmarks

TREND	DATASET	BEST METHOD	PAPER TITLE	PAPER	CODE	COMPARE
	SST-2 Binary classification	SMART-RoBERTa Large	SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization			See all
	IMDb	NB-weighted-BON + dv-cosine	Sentiment Classification Using Document Embeddings Trained with Cosine Similarity			See all
	SST-5 Fine-grained classification	RoBERTa-large+Self-Explaining	Self-Explaining Structures Improve NLP Models			See all
	Yelp Binary classification	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding			See all
	Yelp Fine-grained classification	XLNet	XLNet: Generalized Autoregressive Pretraining for Language Understanding			See all
	MR	byte mLSTM7	A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors			See all
	Amazon Review Polarity	BERT large	Unsupervised Data Augmentation for Consistency Training			See all
	Amazon Review Full	BERT large	Unsupervised Data Augmentation for Consistency Training			See all

Papers with Code

Sentiment Analysis Using Pre-trained Embeddings

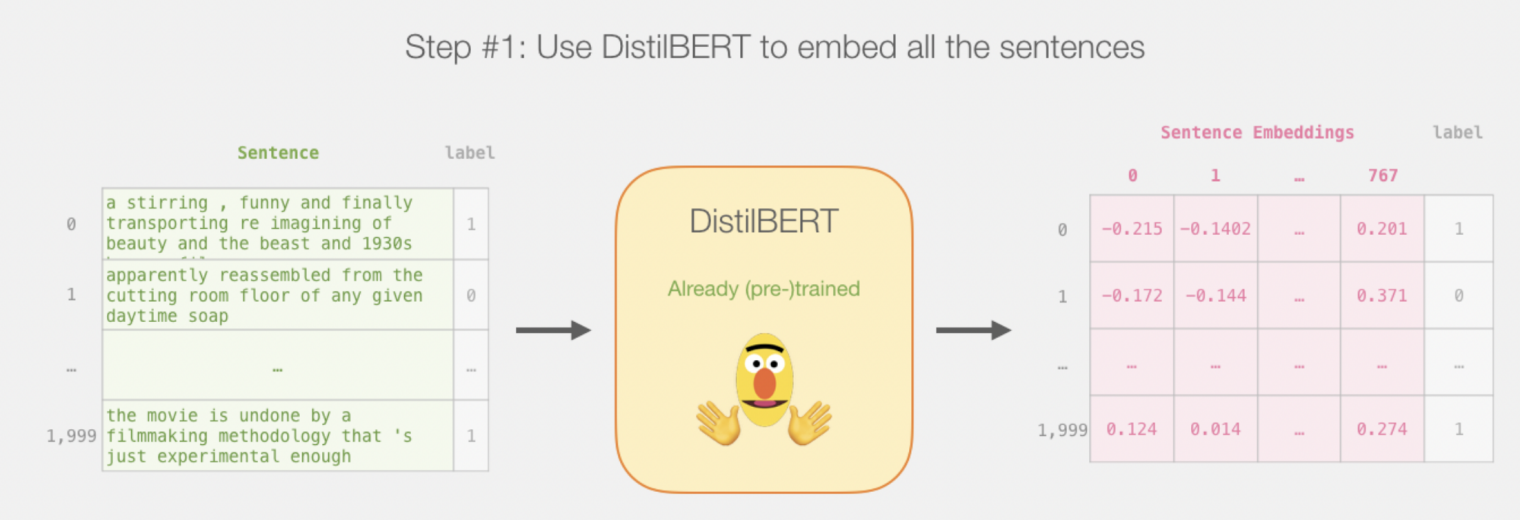
Our goal is to use pre-trained embeddings, only training a linear classifier. This task is made even more lightweight in this example by using DistilBERT.



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-trained Embeddings

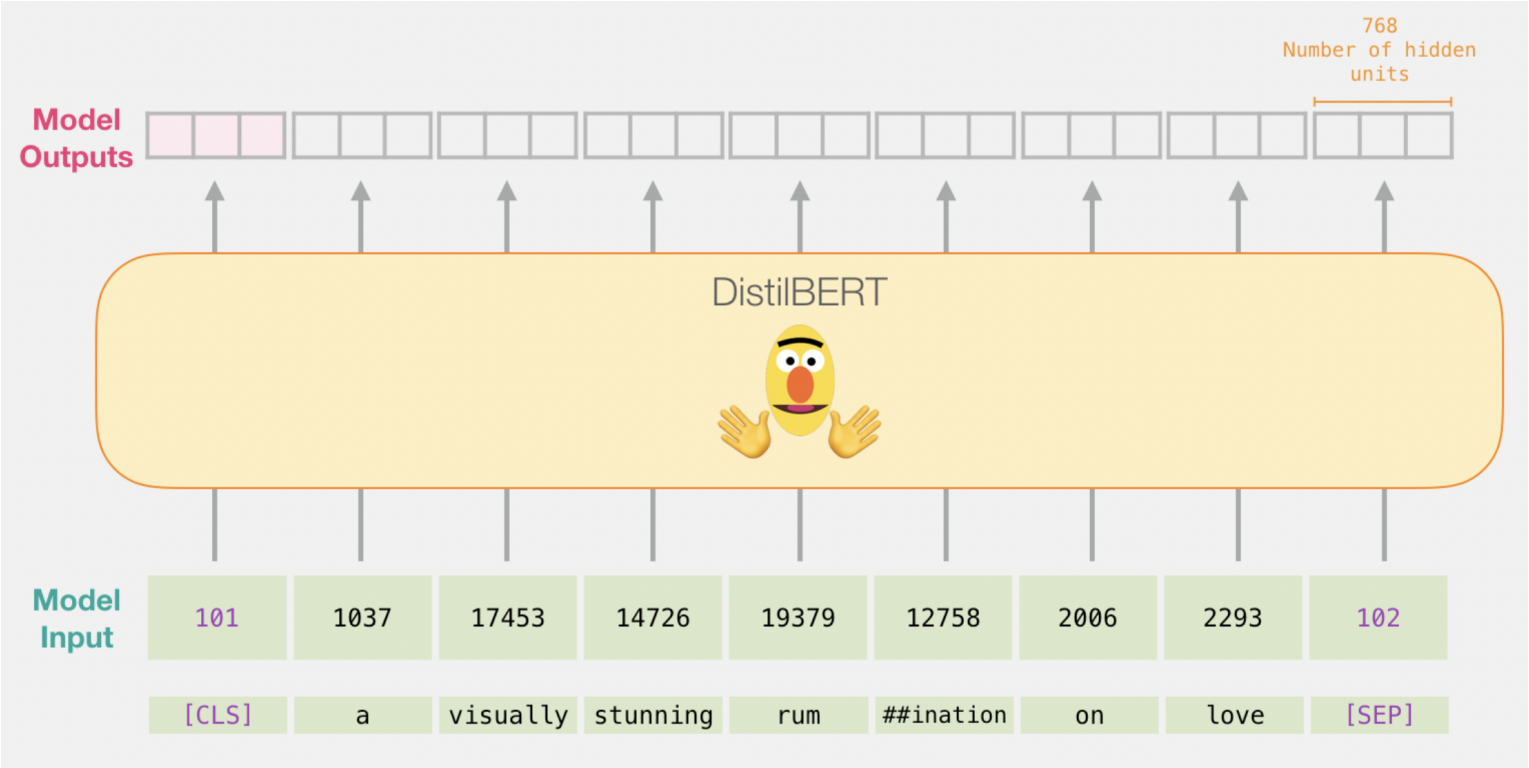
SST2 consists of sentences, and DistilBERT will be used to create embeddings for these sentences.



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

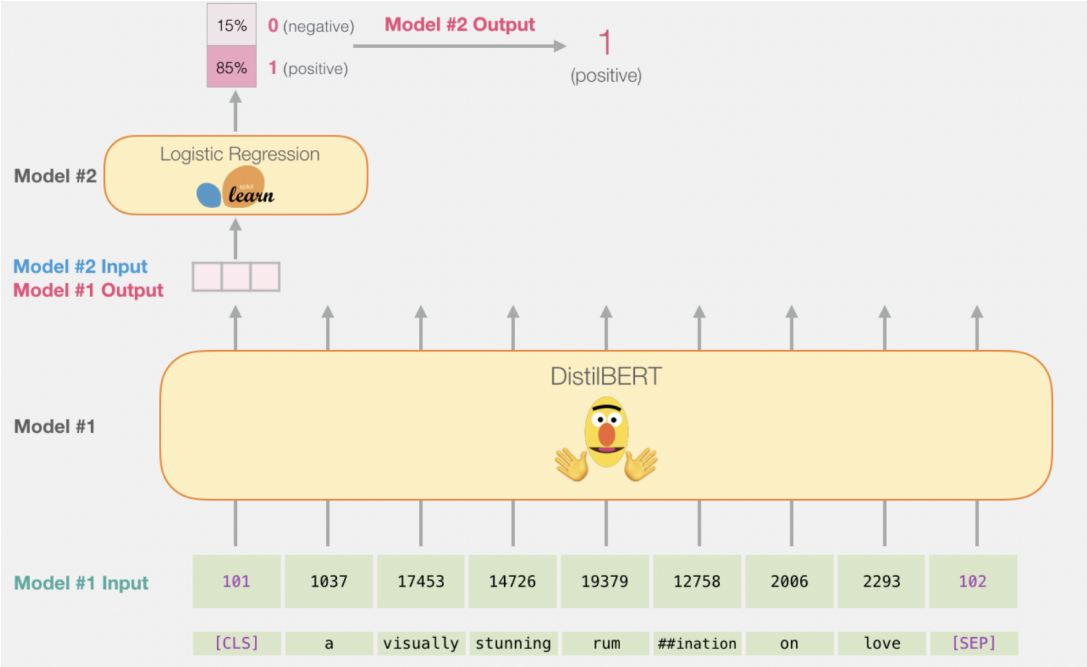
To get the sentence level embeddings, need to place the [CLS] token at the start of the segment you want to embed and the [SEP] token at the end



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

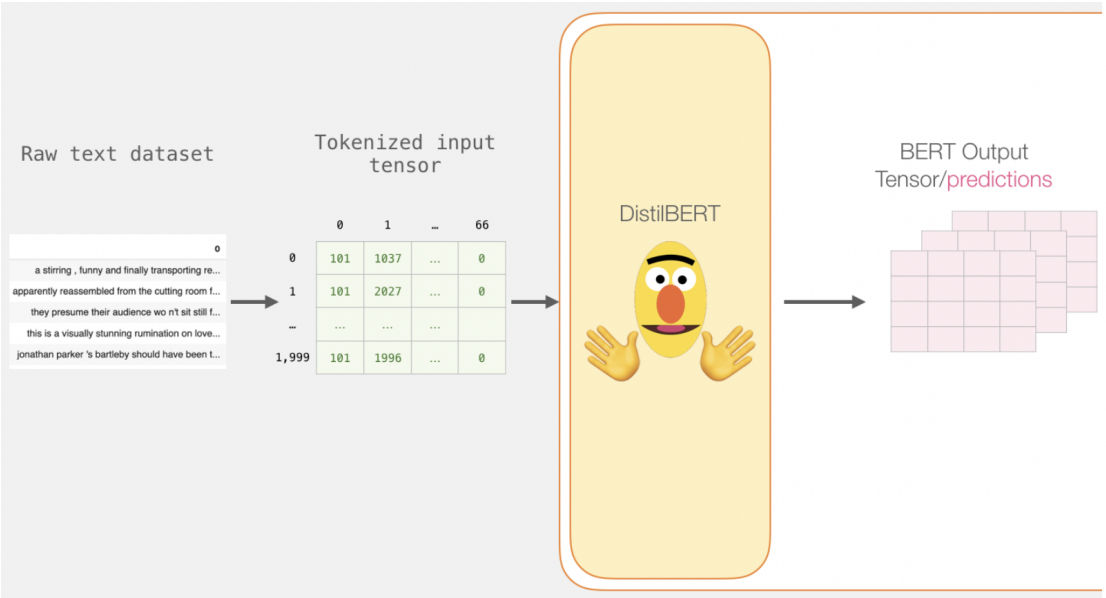
Ignore everything but the embedding for the [CLS] token



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

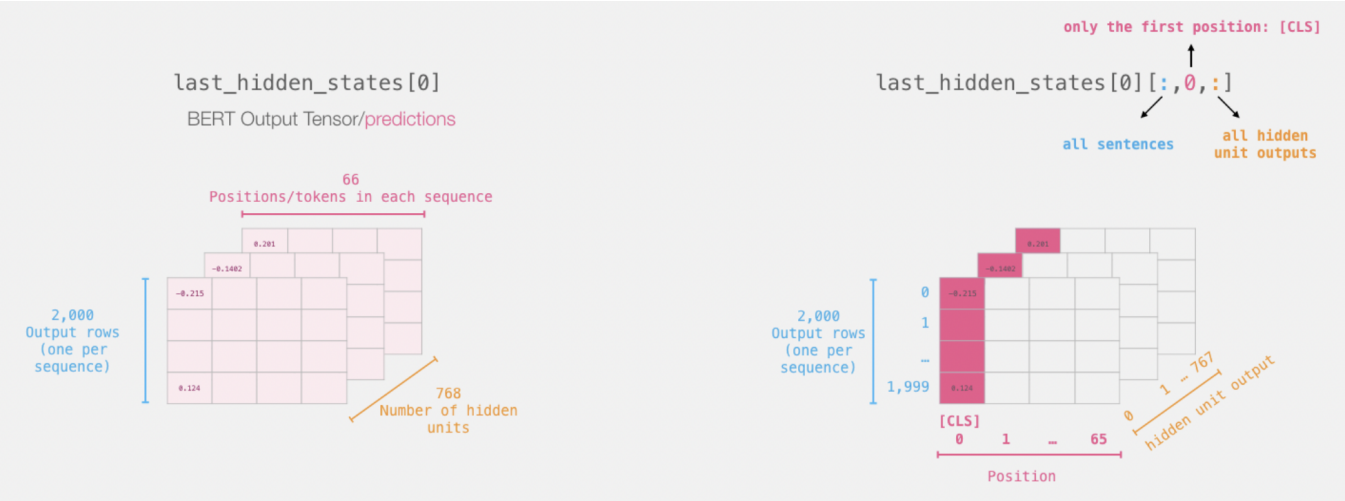
Suppose we want to process 2,000 reviews, and the longest sequence length is 66 tokens. The last hidden state from DistilBERT is a tuple with the shape (minibatch size, max number of tokens, number hidden units) $= (2000, 66, 768)$



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

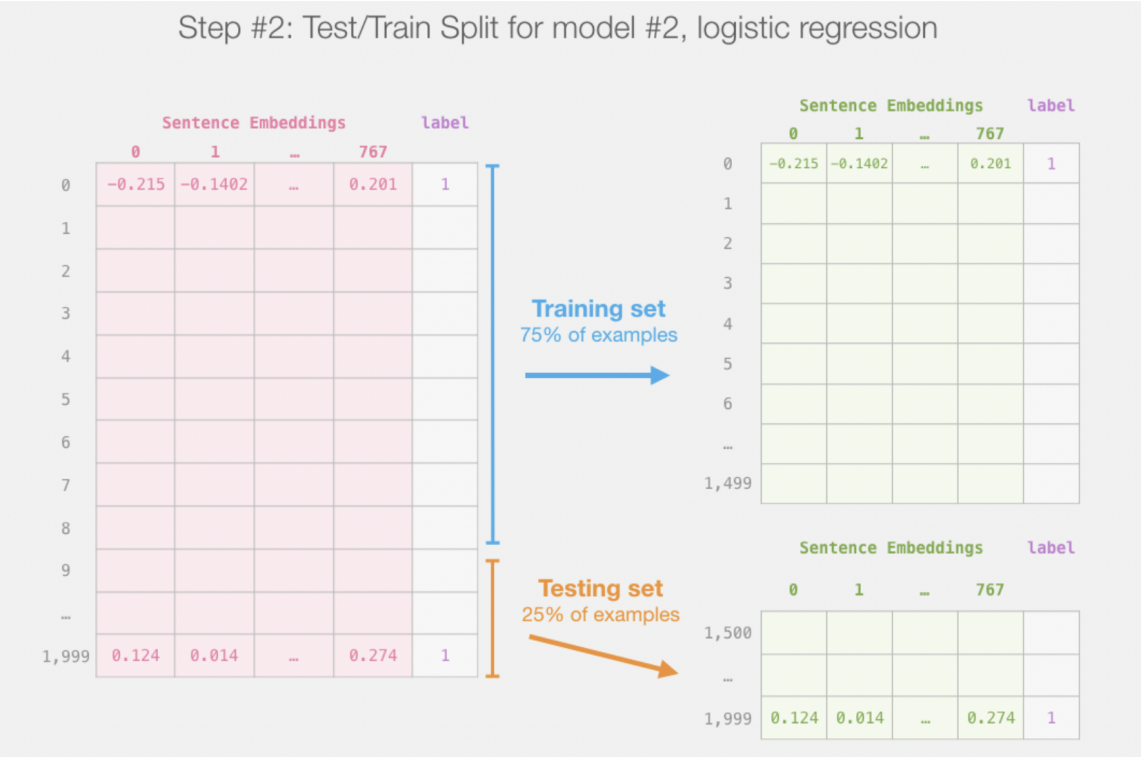
Only need the embeddings from the class token:



https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

Split these data into test and train and then train the linear classifier:



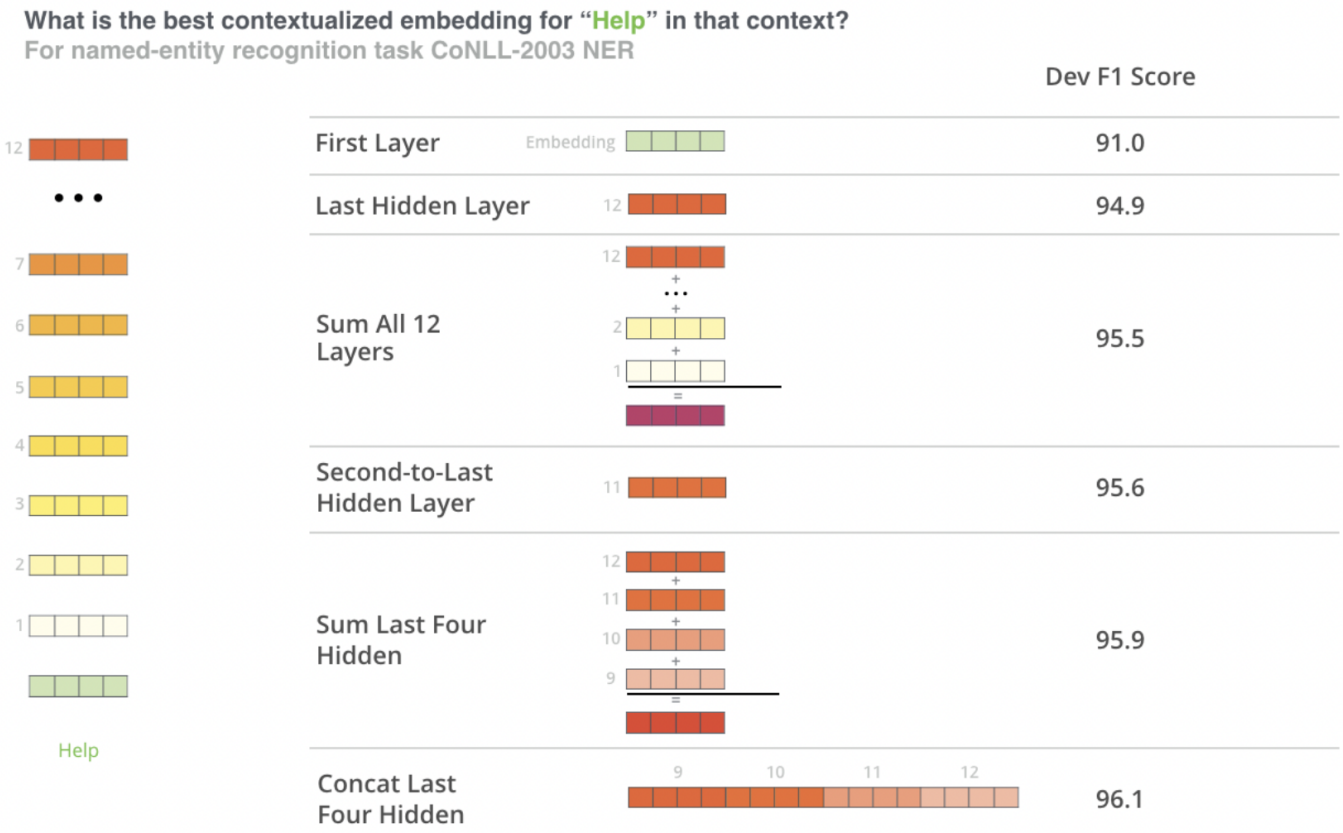
https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Sentiment Analysis Using Pre-Trained Embeddings

- This approach achieves accuracy of 81%
- Fine-tuned DistilBERT achieves 90.7% accuracy
- Fine-tuned BERT Large achieves 94.9% accuracy
- Fine-tuned T5, ALBERT, and RoBERTa now top 97% accuracy (see Papers with Code)

Sentiment Analysis Using Pre-Trained Embeddings

Could probably do better with the cheap approach by thinking more carefully about which hidden representations to use



http://jalammar.github.io/illustrated-bert/