Combining Models

EECS 498

(Remote) Lecture 21

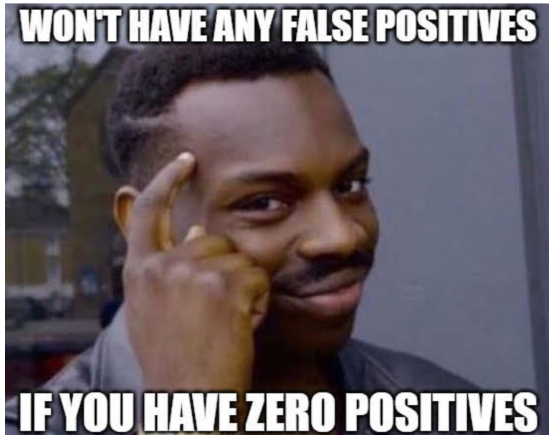
Reminders

- Due next Monday, 4/6
  - PC5 (Cooperative Testing) due 4/6
  - PC6 (Sprint Review 3) due 4/6, delivered as YouTube video
    - Please also upload your raw video to Google Drive (so others can download)
    - SR3: Please review group feedback
- No lectures next week
  - Instead, you will use the time to review each group's SR3 video presentation
- PC7 (Final Presentations) will be a **scheduled telecon** with your team
  - Schedule a 30 minute block here: https://calendar.google.com/calendar/selfsched?sstoken=UVBaMkN5bk9KelVRfGRlZmF1bHR8Mjk4MTllNjJjODMyODdkODk3MzU4YjNmNWIxZDUyNTI
    - Try to have most/all your team members present for that

Review: Evaluating ML Models

- **Model performance** is evaluated with respect to **True Positives, True Negatives, False Positives,** and **False Negatives**
- Evaluated with respect to **binary tasks** over an **evaluation set**
  - *Intent classification*: did the model correctly classify intent X?
    - NB: weight or average performance on a per-intent basis
  - *Slot extraction*: did the model correctly classify a token as slot y?
    - NB: weight or average performance on a per-slot label basis
- We can use TP/TN/FP/FN stats to compute **Precision**, **Recall**, $\boldsymbol{F_1}$ **score**, and **Accuracy**

Review: True and False Positives



- We can compute a **confusion matrix** based on the output of the model for **each utterance** in the **evaluation set**
  - (can be done on a per-intent or per-slot basis, or averaged)

Ground-truth		Predicted in class X	Predicted not in X
Actually **in** class X	50	45 (True Positives)	40 (True Negatives)
Actually **not** in class X	50	5 (False Positives)	10 (False Negatives)

- From the confusion matrix, we can compute Precision, Recall, and Accuracy scores

Review: Precision, Recall, F_1, Accuracy

- These scores characterize the mistakes made by a classification model
- Precision
  - Fraction of actual in-class values compared to all predicted in-class values
    - TP / (TP + FP), also called the Positive Predictive Value
- Recall
  - Fraction of predicted in-class values compared to all actual in-class values
    - TP / (TP + FN), also called the Sensitivity
- F_1 score
  - Combination of precision / recall to account for both types of error
    - p*r / (p + r) = TP / (TP + FP + FN)
- Accuracy
  - Fraction of correctly classified vales over all classifications
    - (TP + TN) / (TP + TN + FP + FN)

Review: Model Evaluation Considerations

- **Slot Extraction**: Train by labeling portions of utterance
  - Yo fam get me a burger.
  - O B:person O B:person O B:food

- In larger utterances, most tokens are O
  - Do we care as much as Precision/Recall for O tokens?
  - Consider: is identifying whether a token is *any* slot the same as identifying its *slot label*?

- **Remember:** the true and false positives and negatives *mean* something in the context of your task. Don't blindly apply statistics.

Review: Datasets and Overfitting

- When evaluating models, we practice a discipline notion of diving datasets
  - Training set — Utterances used to compute weights in NN
  - Development set — Utterances used to fine-tune the NN and debug
  - Evaluation set — Utterances used to evaluate performance (e.g., F1)

- It is **critical** that these datasets do not overlap
  - We risk **overfitting** to the training data
  - A model is not useful if it's *only* super good at classifying training data…

Overfitting in Conversational AI

- Virtual assistants can become overfit:
  - Intent Classification
    - Crowdsourcing: insufficient diversity from prompt biasing
      - e.g., Banking Assistant: what if you only ask for rephrasals that use the work “checking”?
    - Insufficient scoping leads to classification failures
      - e.g., Banking Assistant: what if you only think of checking/savings, but not about IRA accounts?
  - Slot Extraction
    - Insufficient token diversity
      - e.g., Banking Assistant: cheques, checks, savings, money market, CD
    - Insufficient context diversity
      - e.g., Banking Assistant: what if all utterances are of the form “… from X to Y…” instead of “to Y from X”? (slot ordering may be overfit)

Review: Word Embeddings

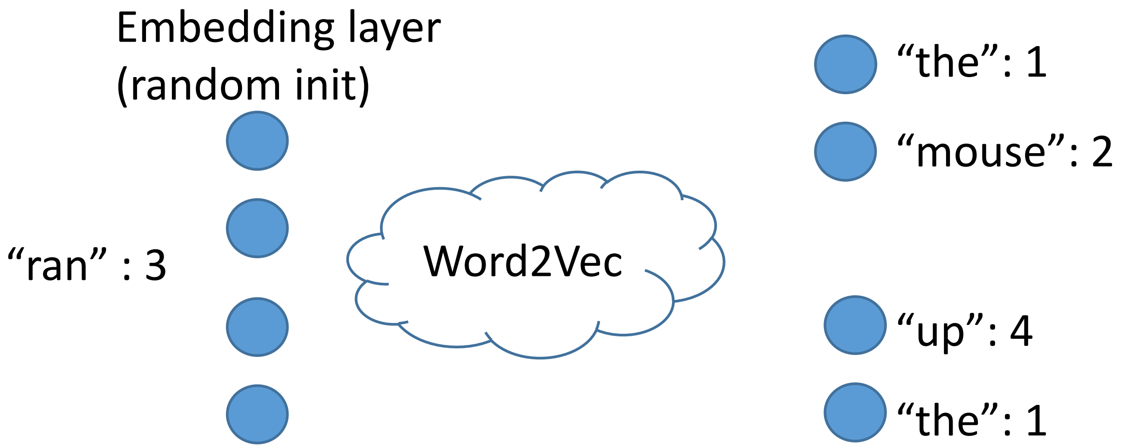
- Embeddings are **compact**, **semantics-preserving** vector representations of individual **words** (or other NLU elements)
- Embeddings are derived from Neural Network weights near the **input layer** (called an **embeddings layer**)
- Example: Word2Vec is trained to predict surrounding words given an input word in a sentence

The mouse ran up the clock.

→

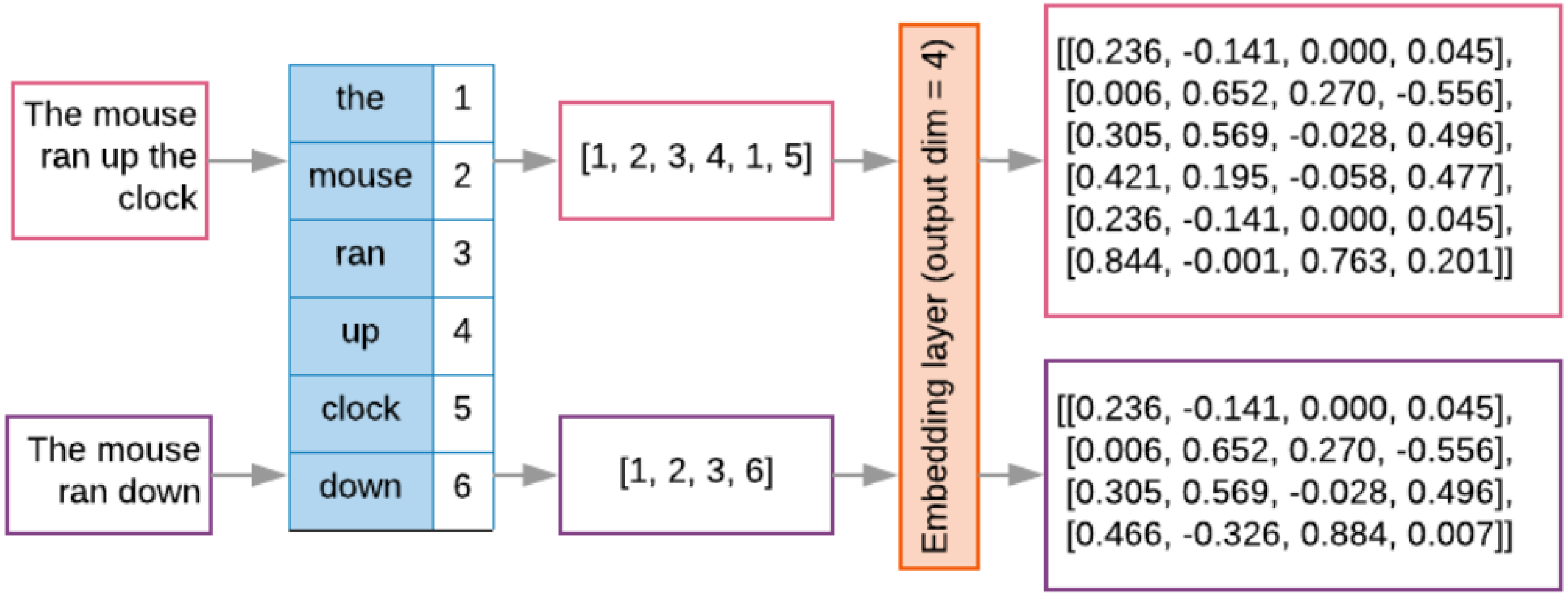
[1, 2, 3, 4, 1, 5]

The	1
Mouse	2
Ran	3
Up	4
Clock	5
Down	6



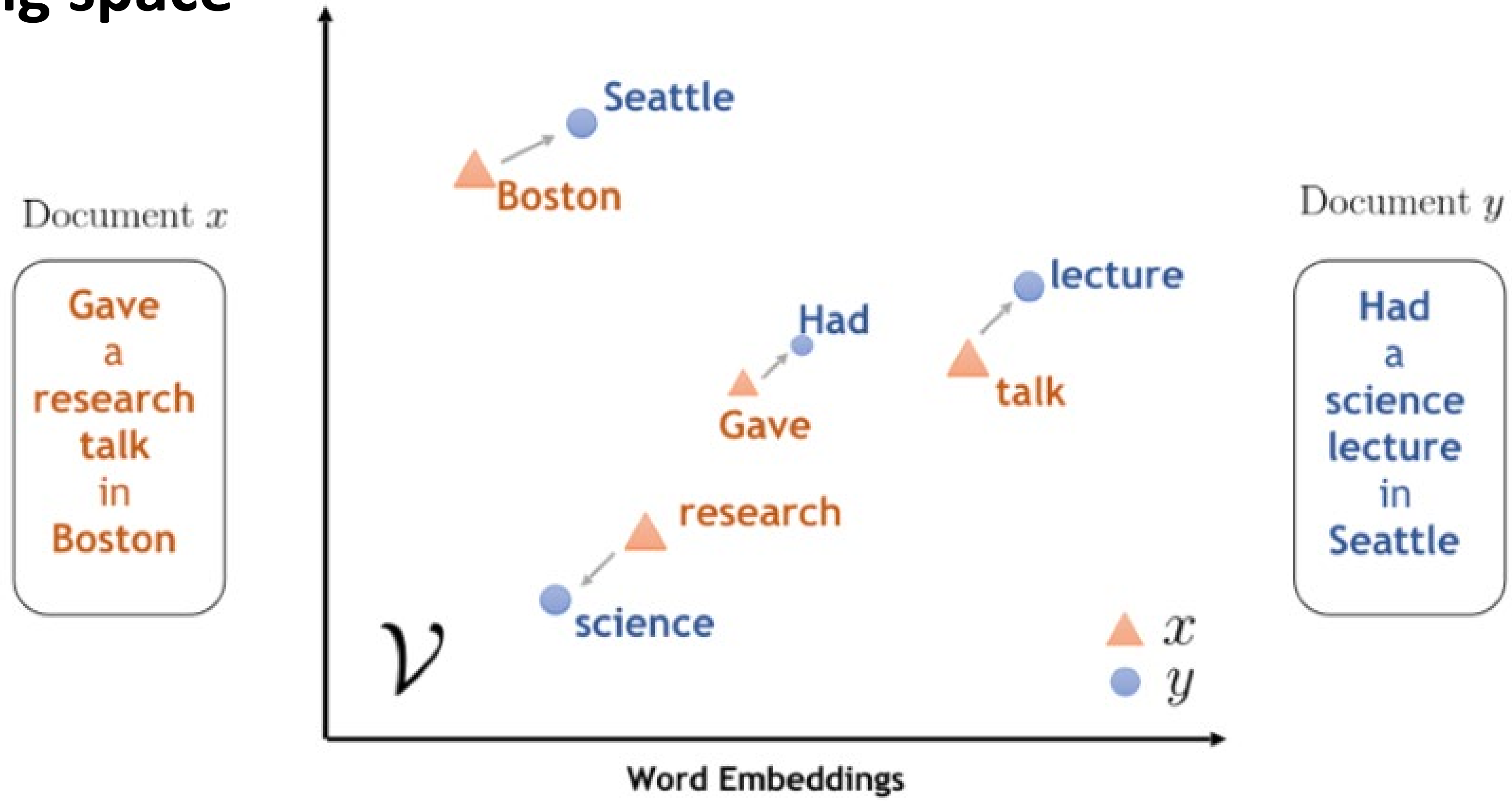
Review: Word Embeddings

- Embeddings can capture **semantic relationships** between words
  - e.g., for Word2Vec, the network learns words that **frequently co-occur** within some small 5-word **span**
- **Dimensionality** depends on the size of the embeddings layer



Review: Word Embeddings

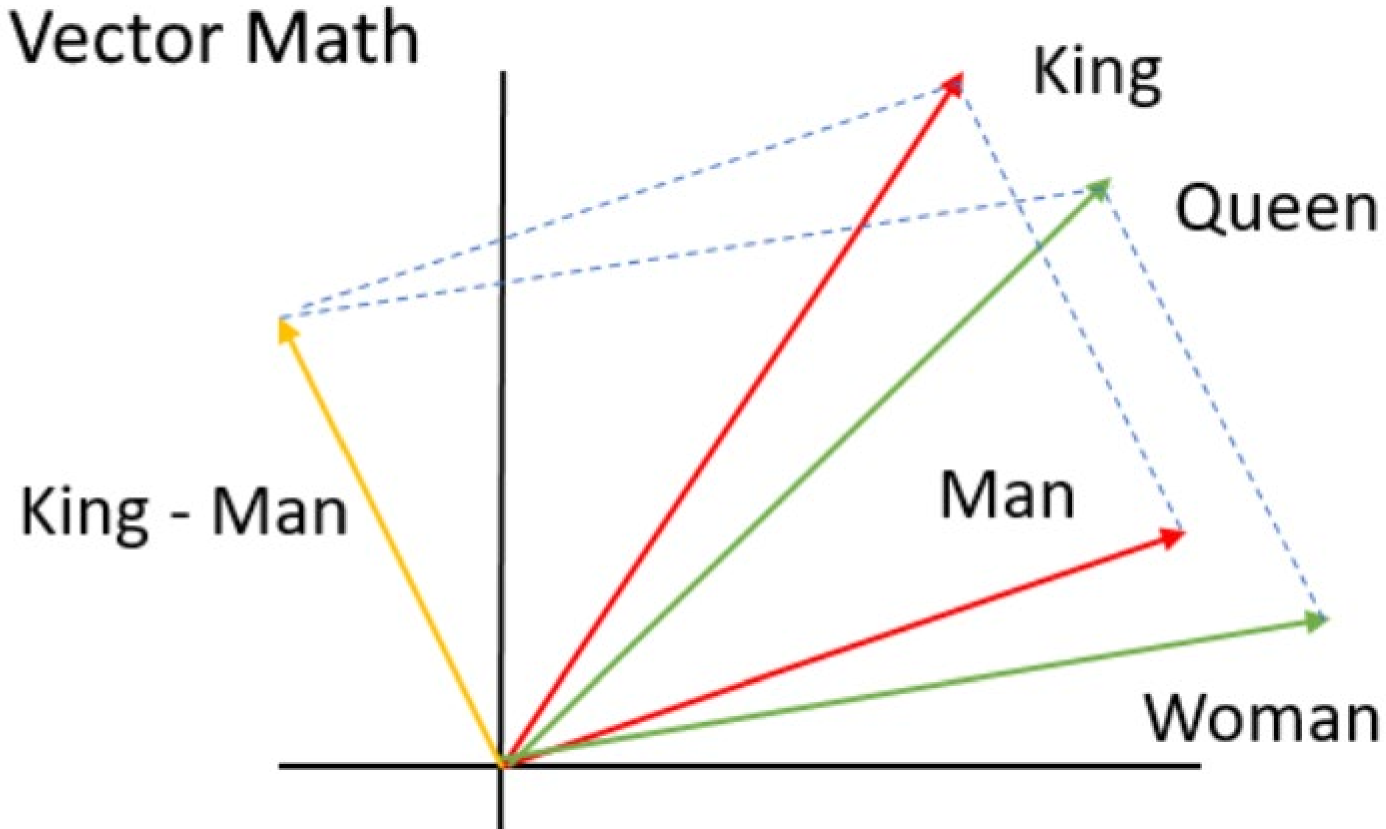
- Words that are **related semantically** should be **close** in the **embedding space**



Review: Word Embeddings

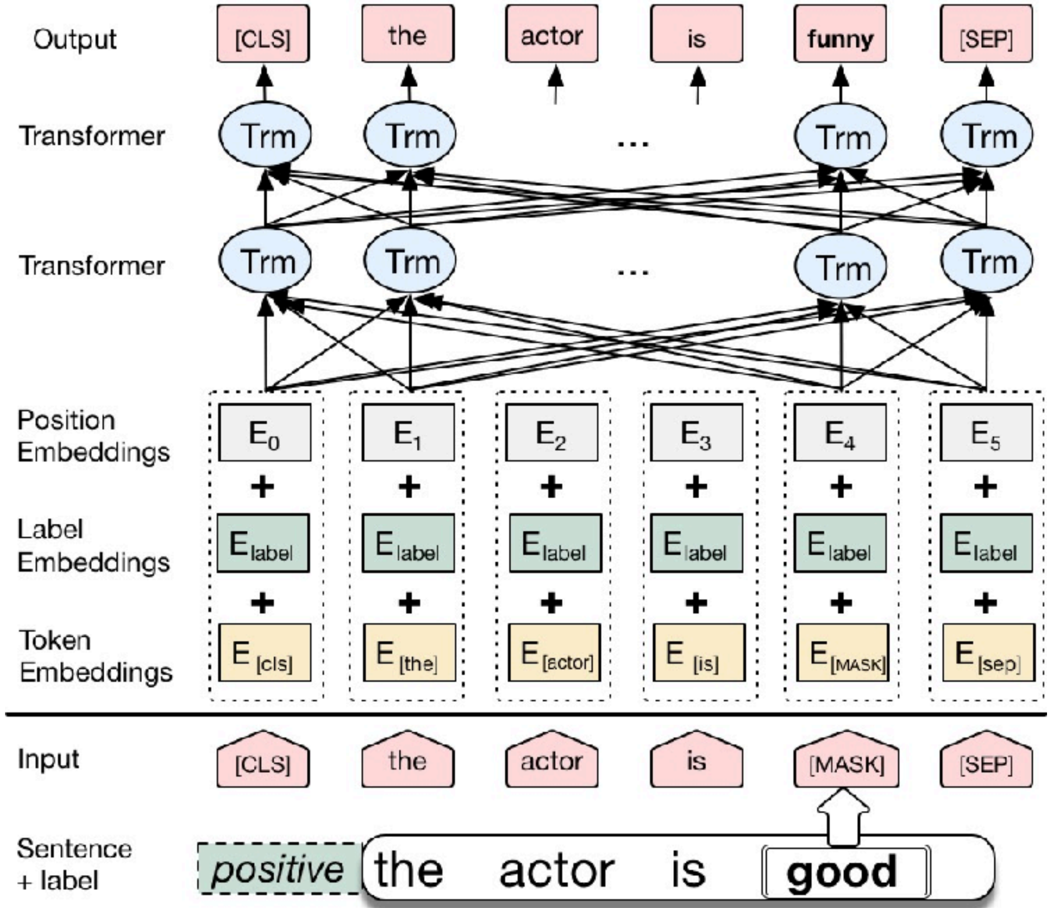
- Once we move into the **embedding space**, we desire arithmetic properties that **preserve** semantic relationships

- Note: King – Man + Woman = Queen



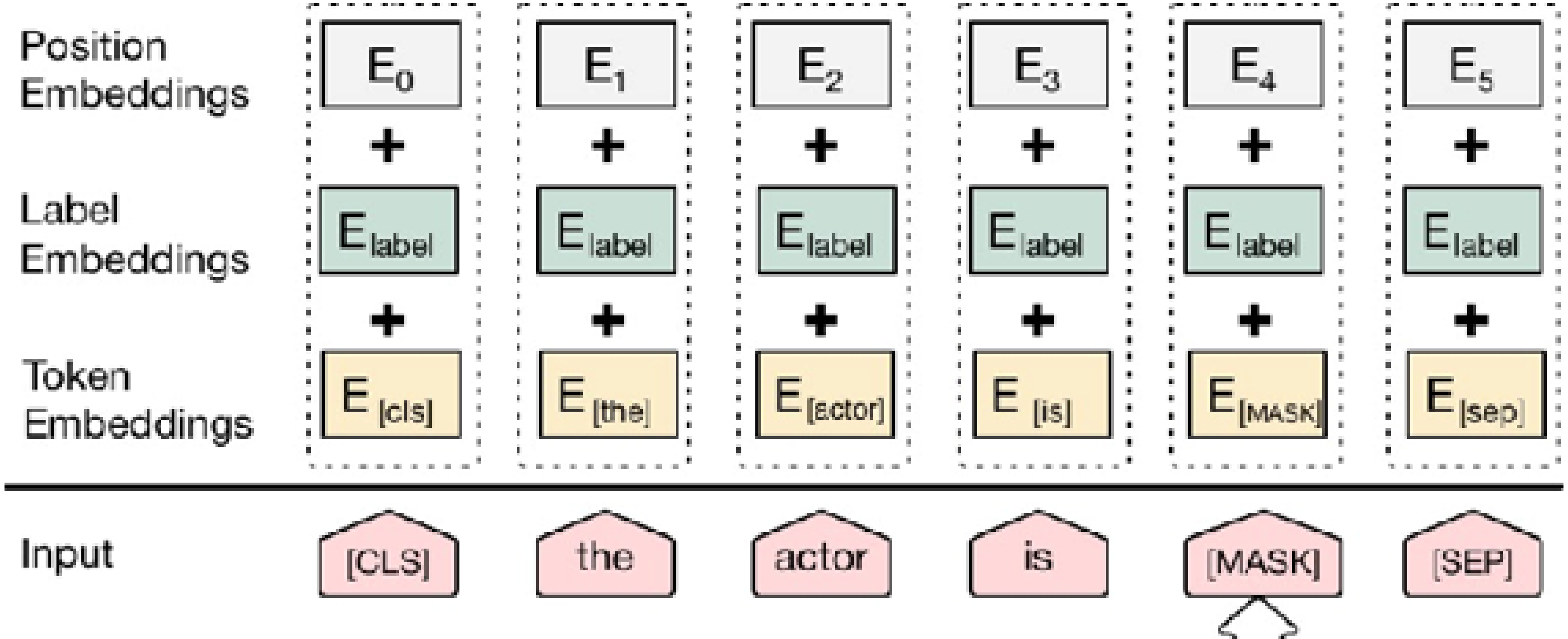
Review: Bert

- Bert is an advanced language model from we can derive **contextual embeddings**



Review: Bert

- Input representation consists not only of token-level embeddings, but also position and label embeddings
  - Allows embeddings to **capture context** (position relative to other tokens) and **semantics** (the embeddings must 'learn' to compensate in the presence of a MASK



One-Slide Summary: Model Combinations

- Bert is itself a combination of many pieces… How does it work?
  - Attention / Transformer
  - WordPiece Vocabulary

- NLU pipelines consist of **Intent Classification** and **Slot Extraction**
  - Slot Mapping comes later, but may or may not involve a model
  - Downstream or end-to-end performance can be **very** different from individual model performance

A Deeper Dive on Bert

- Bert uses **WordPiece** vocabulary as the input representation
  - WordPiece represents pieces of words in sequence

```
conversational artificial intelligence:
['conversation', '##al', 'artificial', 'intelligence']
conversationl artifcial intelligence:
['conversation', '##l', 'art', '##if', '##cial', 'intelligence']
```

  - Pieces of words are mapped to unique vocabulary identifiers (i.e., numbers)
  - Allows *some* robustness against misspellings
    - RoBERTa takes this a step further
    - FastText is another representation for robustness against misspellings

A Deeper Dive on Bert

- **WordPiece** is an example vocabulary that attempts handling some amount of **out-of-vocabulary** tokens
  - **Consider:** words like “Big Mac” or “Deloitte” may not directly map to typical English words… what if they aren’t present in our vocabulary at training time?
  - **Recall:** Lexical analysis. How do we break up tokens in an utterance?
    - Morphological normalization can help reduce vocabulary size
  - WordPiece uses Subwords: frequently-occurring sequences of characters

antelope

ants elope

cantaloupe

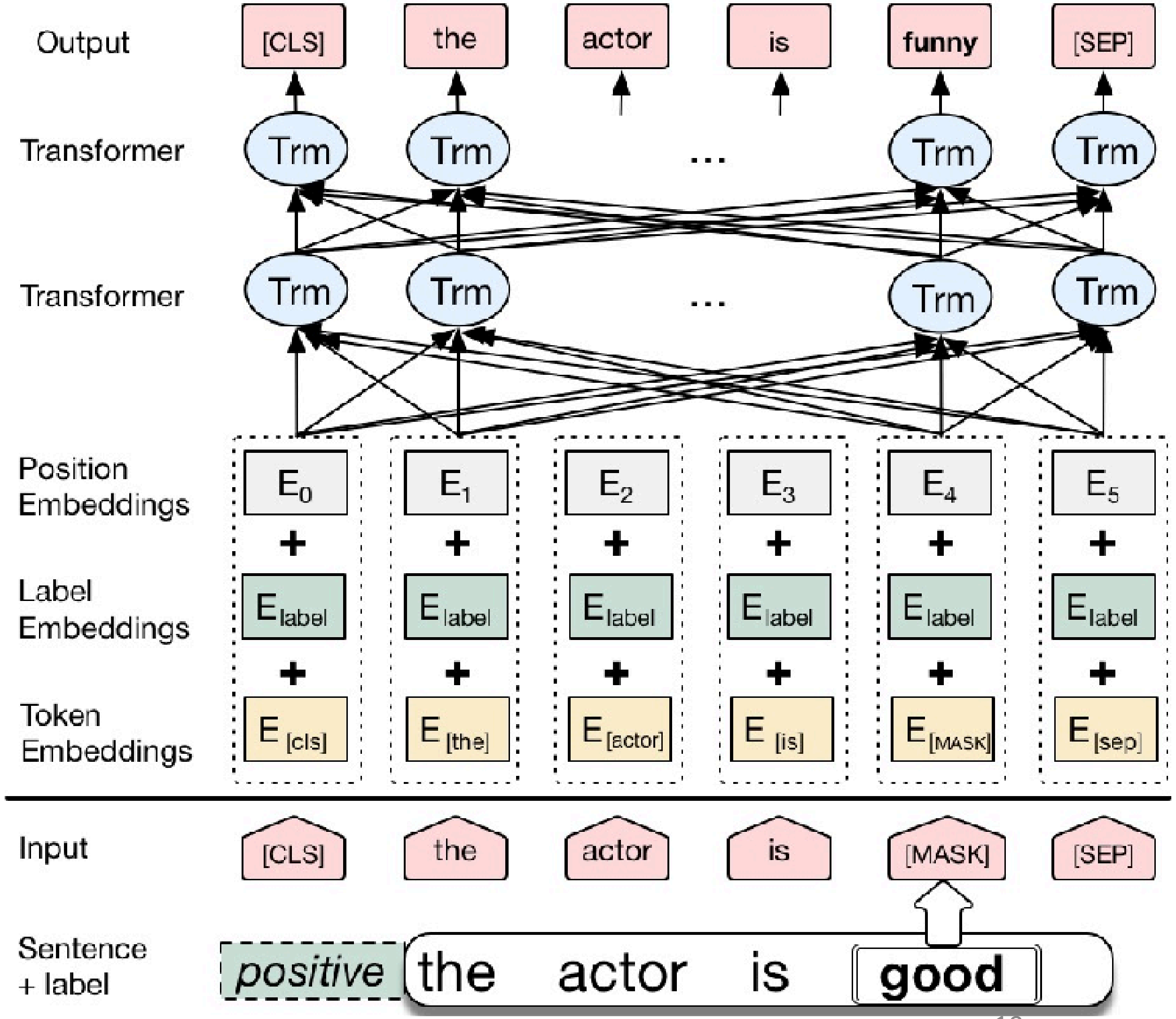
A Deeper Dive on Bert

- Bert: Bidirectional Encoder Representations from Transformers
  - **Bidirectional**: Bert represents a **language model** that works in both directions
    - i.e., left-to-right and right-to-left.
    - e.g., Predict X in "… X jumps over the lazy dog" <- only has right-sided context
    - Bert can **learn** from **both left-** and **right-sided** context in input sequences
  - **Encoder**: Basically the same thing as an embedding
    - *Technically,* encoders encompass all layers that lead up to the embedding
  - **Representations**: a method for representing data
    - An Encoder Representation is like an embedding
  - **Transformers**: A type of neural architecture that applies well to NLP
    - Also, robots in disguise

Bert: Bidirectional

- The architecture of Bert allows it to *learn* from context on both sides of each token
  - **Contextual embeddings**

- **Transformers** enable this behavior
  - For each word, the NN accounts for the *attention* it should give to all other words in the input



Bert: Encoder Representations

- An **Encoder** is a NN is produces **embeddings**
  - In the case of Bert, this produces a robust **contextual representation** by accounting for
    - Positional information of each token (i.e., is it the 1st, 2nd, 3rd, etc. word in the sentence?)
    - Encoding information from other tokens in the sequence
      - As the model trains, these encodings learn from contexts in which the tokens appear

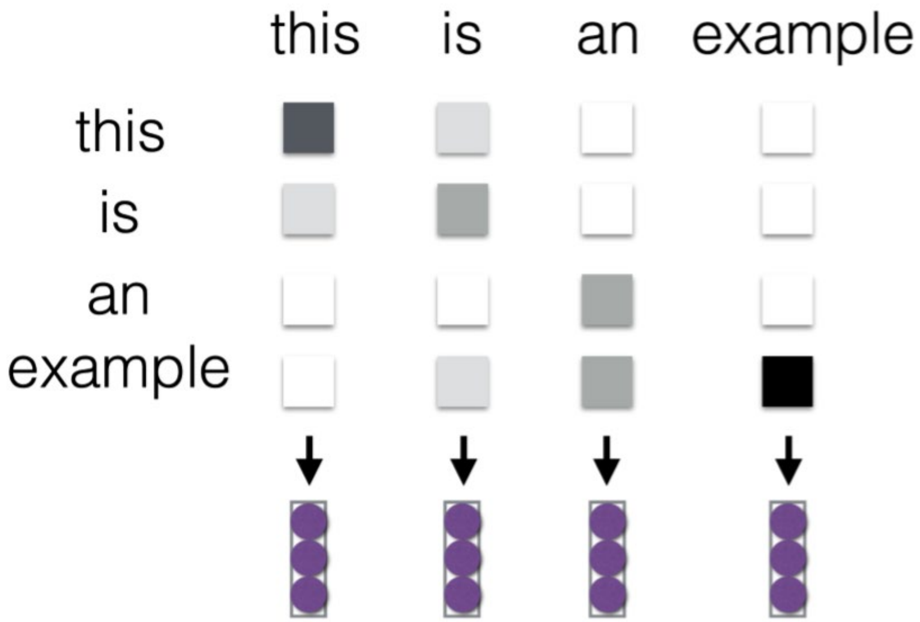
- In particular, Bert's Encoders use **self-attention**
  - **Attention** is a formal notion of relative importance
  - **Recall:** RNNs consider each word at a time – each inference step (usually) has limited information about other tokens
    - The Attention mechanism allows learning a representation of importance
      - "The cat ate its food." <- Attention learns: "cat" important for "its"; "ate" important for "food"

Bert: Encoder Representations and Attention

- Bert uses a **self-attention** mechanism
  - Attention consists of a separate NN (a sub-graph of Bert as a whole)
    - The NN learns **relative importance** of words in a **sequence** over a whole **corpus**
    - The NN can *attend* to both left and right context (*bidirectional*)

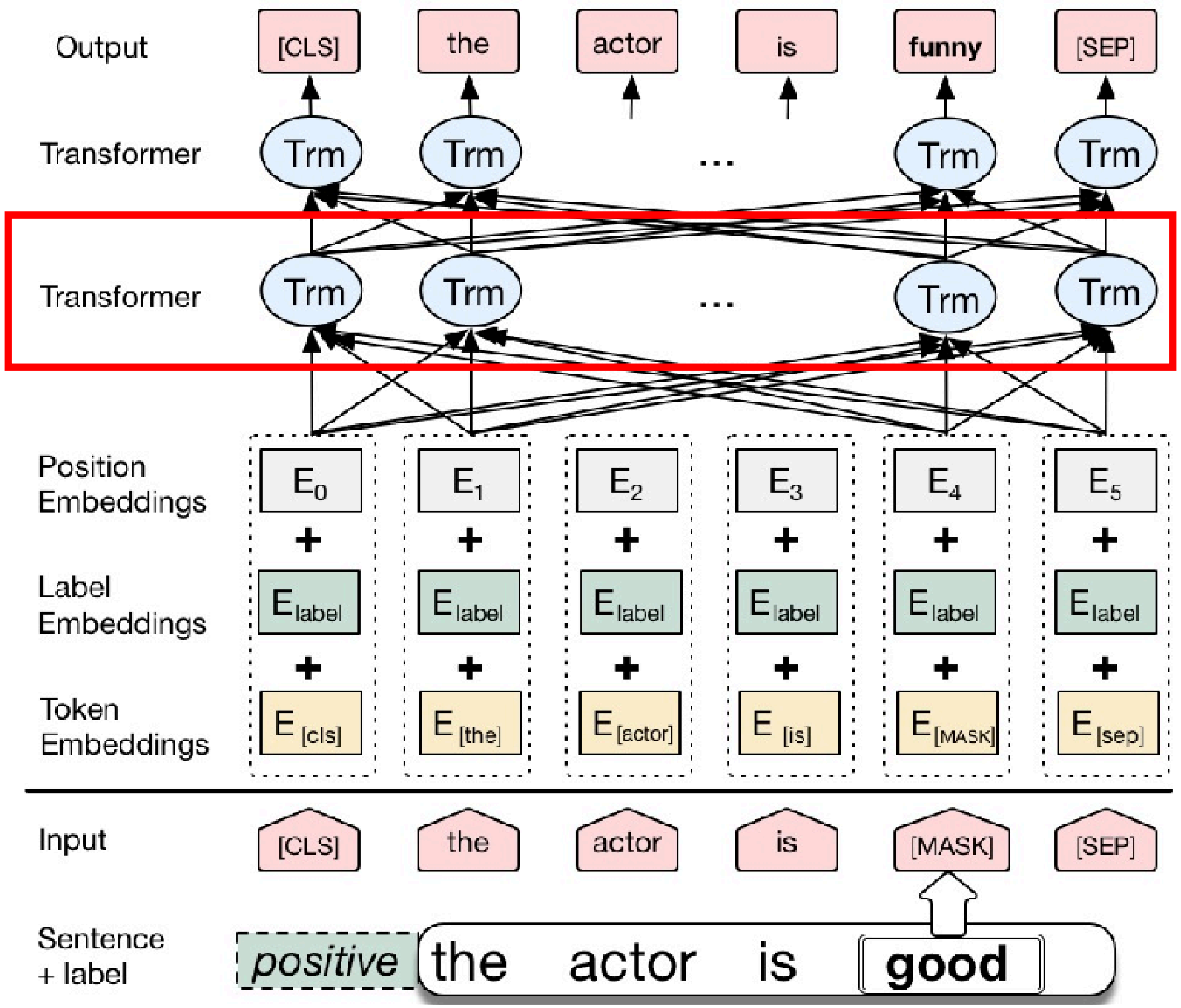
    - Predicting "this" doesn't really depend on "an" or "example"
      - "is" is kind of important
    - "example" depends on "an" and "is" -> hinting at correct sequence of words

  - Attention critically important in NLP!
    - **Translation** makes use of attention



Bert: Attention Explained More

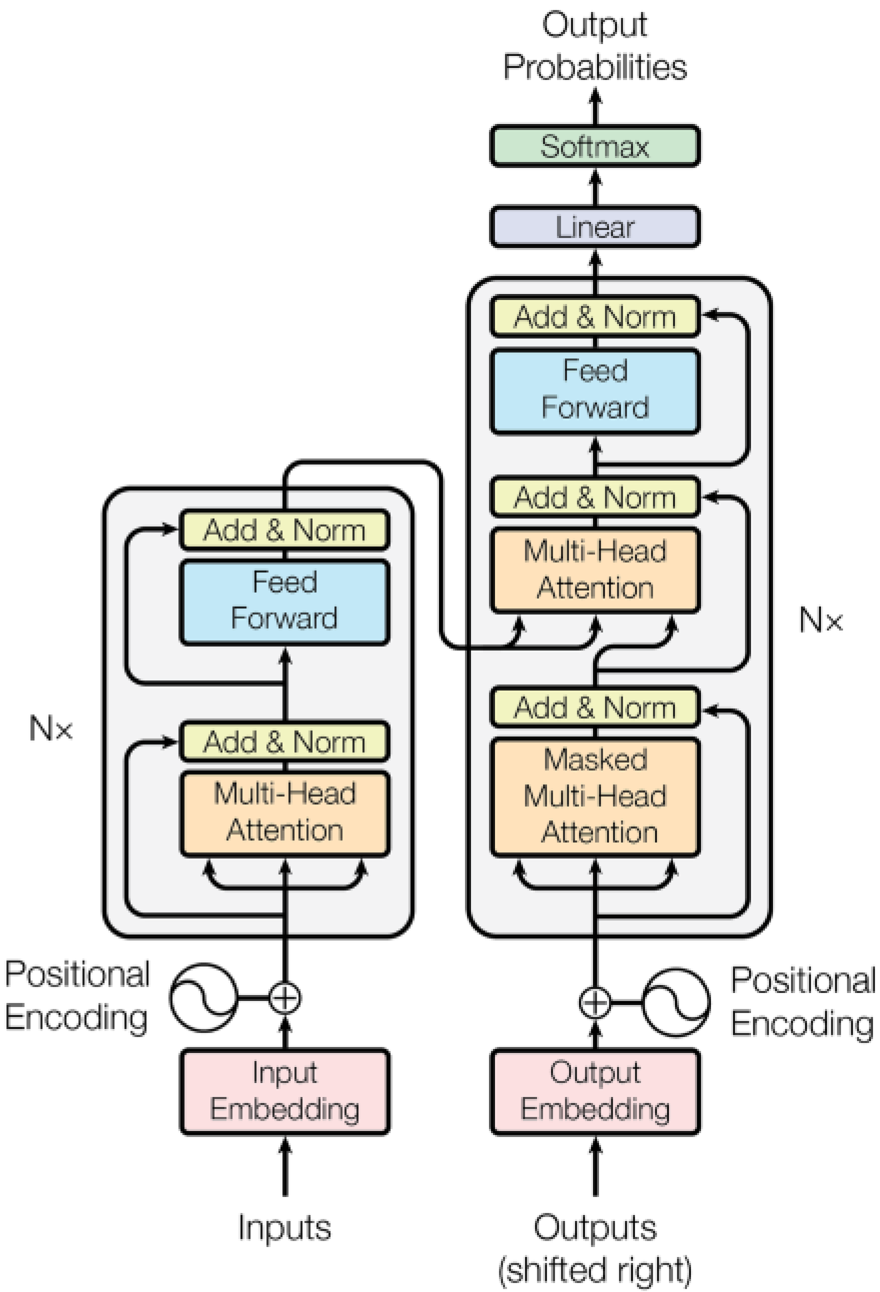
- The self-attention mechanism causes learned embeddings to reflect attention
  - That is, the **embeddings** for **one token** are forced to **account for** the **attention** given **to other tokens** in the sequence
- The first Transformer (the Encoder) is fed input from all tokens in the sequence
  - Self-attention allows it to encode while modulating for relative importance



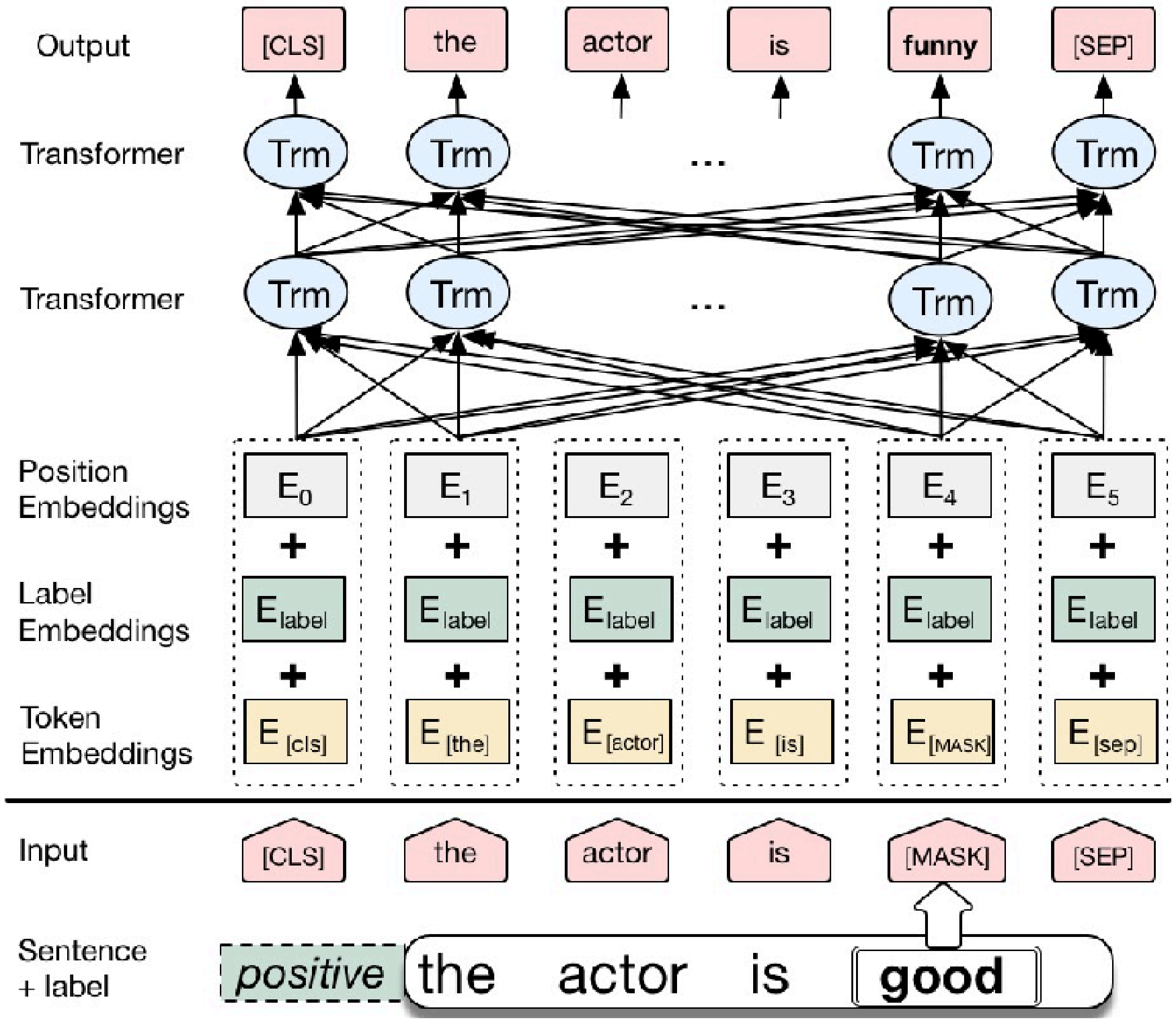
Bert: What is a Transformer?

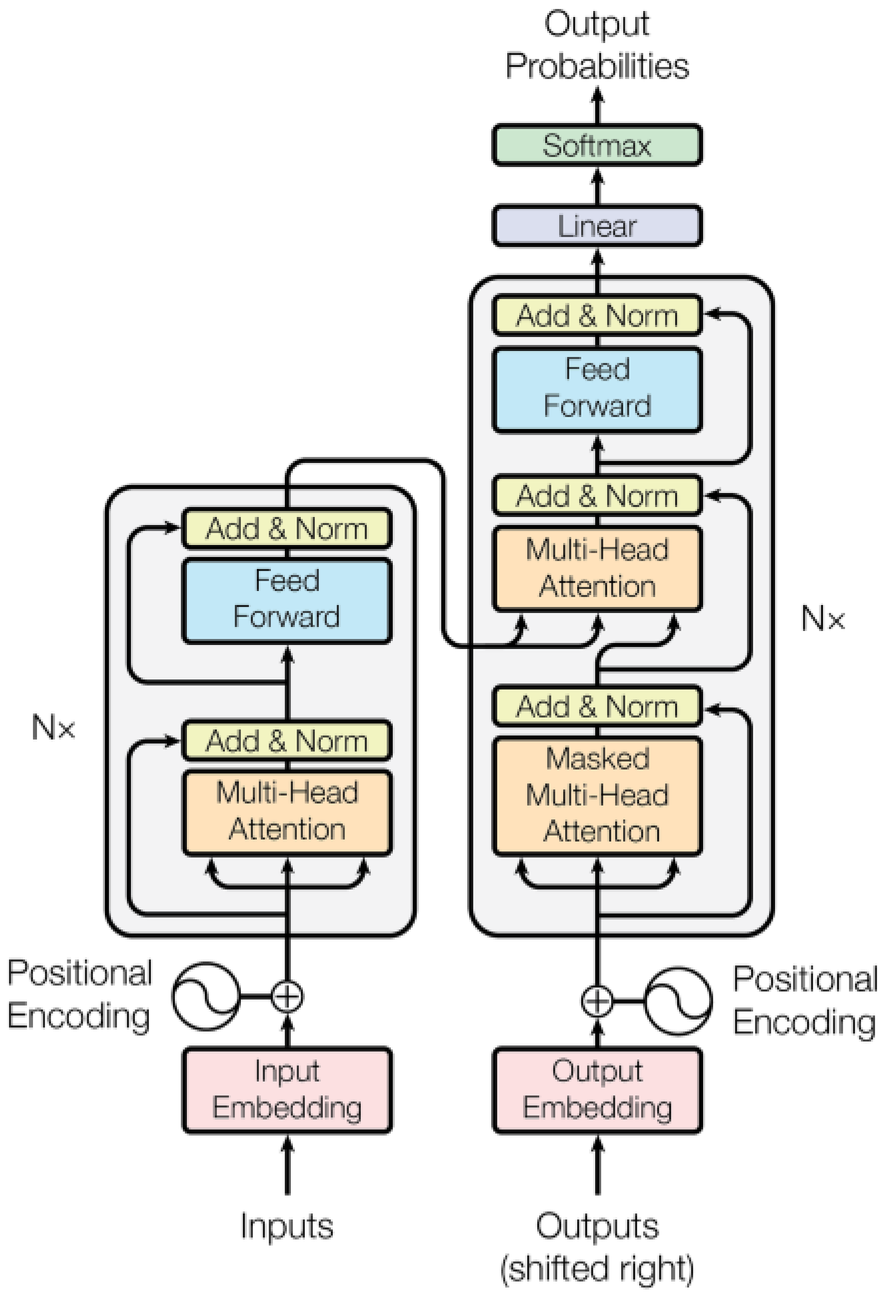
- Bert uses **Transformers** to help learn context from sequences

- A **Transformer** consists of an **Encoder** and **Decoder** with **self-attention**
  - **Recall:** Encoder is a NN that produces some embedding
  - **Decoder**: turns an embedding vector into a vocabulary identifier
  - **Attention**: a sub-NN that allows learning relative importance of tokens in a sequence



Bert: Transformers





Bert: Summary

- We have discussed Bert as a mechanism for acquiring robust contextual embeddings
- In practice, Bert can do a lot more
  - The word embeddings were more of a nice "side effect" of the architecture
  - **Sentence Prediction**
    - Given one sequence of words, predict the next sequence
  - **Question-answering**
    - Learns relationships between question sequence inputs and answer sequence outputs
- Bert is unwieldy
  - 11GB of VRAM to run?

Bert: Semantic Entailment

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
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

Sentence: The car honked down the road.
Label: Unacceptable

Bert: Logical Analysis

```
A girl is going across a set of monkey bars.  She
(i)  jumps up across the monkey bars.
(ii)  struggles onto the bars to grab her head.
(iii)  gets to the end and stands on a wooden plank.
(iv)  jumps up and does a back flip.
```

- Run each Premise + Ending through BERT.
- Produce logit for each pair on token 0 (`[CLS]`)

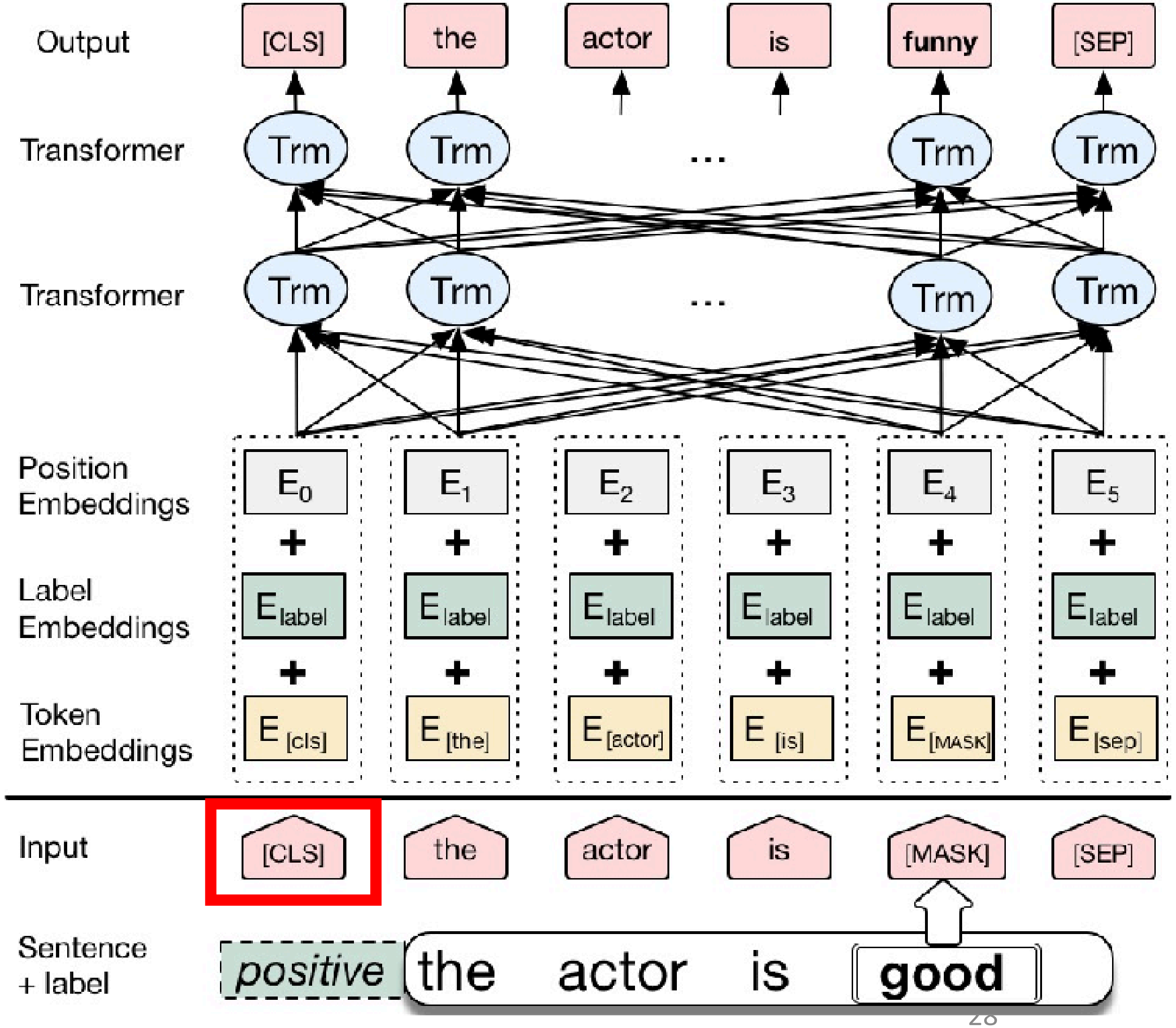
Leaderboard

Human Performance (88.00%)
Running Best
Submissions

Rank	Model	Test Score
1	**BERT (Bidirectional Encoder Representations from Transfo...** *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova* 10/11/2018	86.28%
2	**OpenAI Transformer Language Model** *Original work by Alec Radford, Karthik Narasimhan, Tim Salimans, ...* 10/11/2018	77.97%
3	**ESIM with ELMo** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/30/2018	59.06%
4	**ESIM with Glove** *Zellers, Rowan and Bisk, Yonatan and Schwartz, Roy and Choi, Yejin* 08/29/2018	52.45%

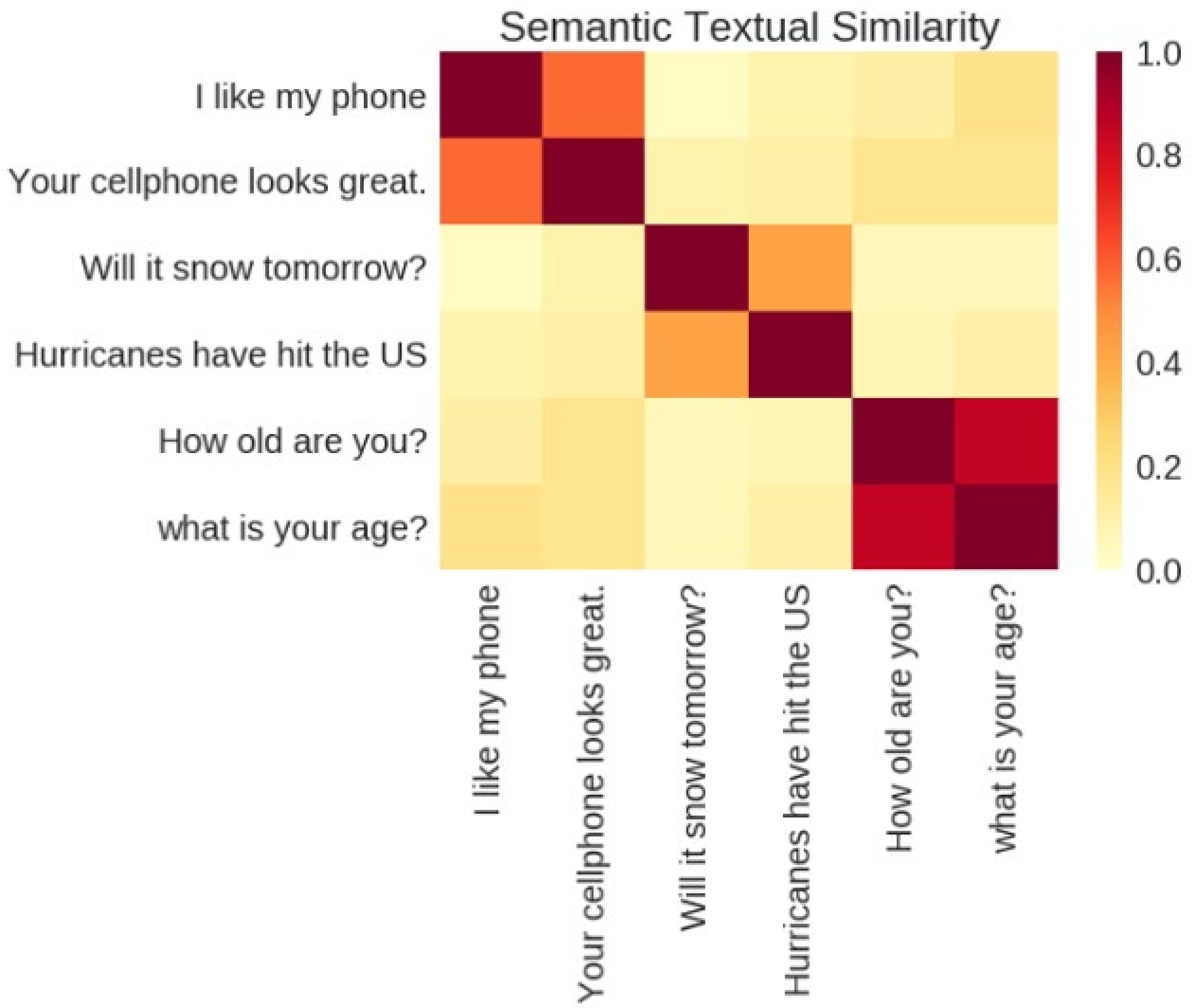
Bert: Sentence Embeddings

- The [CLS] token is meant to represent the start of a sentence
  - **Consider:** The model supposedly learns context in part from position
  - Every sentence "starts" with [CLS]
- No matter what sentence is given, [CLS] always involves context learned from every other word
  - Thus, the embeddings for [CLS] are a **rich representation** of the **whole sentence**



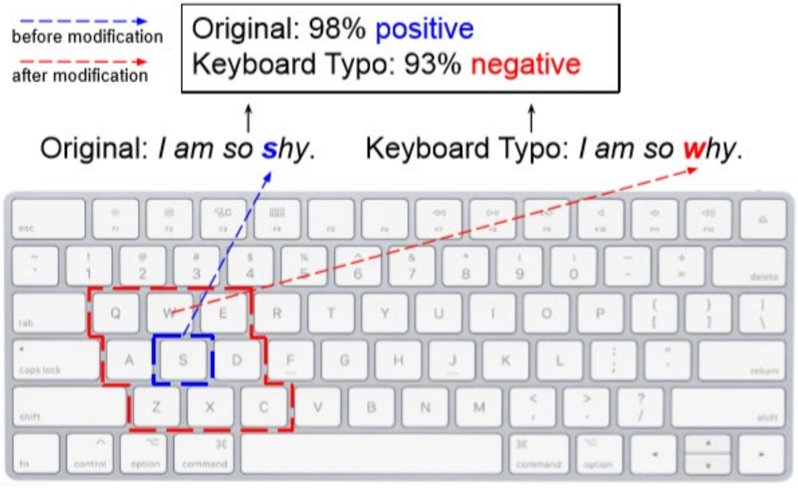
Sentence Embeddings in General

- Embed sentences into vector space
- Useful for comparing sentences semantically
- Word embeddings are used in addition to positional information



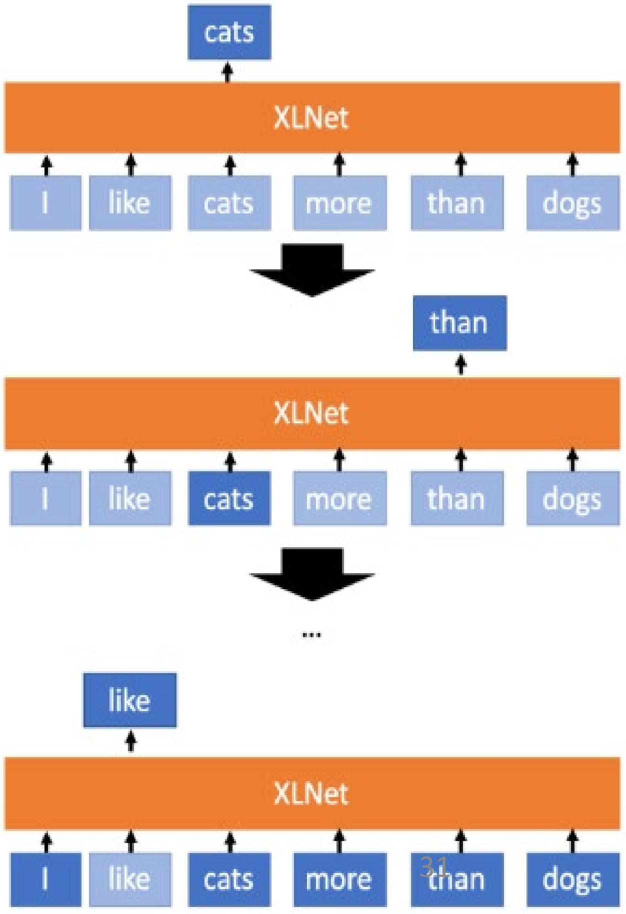
Bert: Shortcomings

- Bert's language modeling assumes **independence** among MASK tokens
  - **Recall:** Bert operates by MASKing some tokens, forcing the embeddings to reflect **context**
  - **Problem**: if multiple MASK tokens appear in a sentence, their ordering and relationship are assumed irrelevant by BERT
    - "I have to fly from MASK_1 to MASK_2" <- wouldn't make sense if the MASKed tokens were "Ithaca" and "Syracuse"
- Bert's input leverages WordPiece
  - **Problem**: Limited robustness against misspellings



XLNet: Even more state-of-the-art?

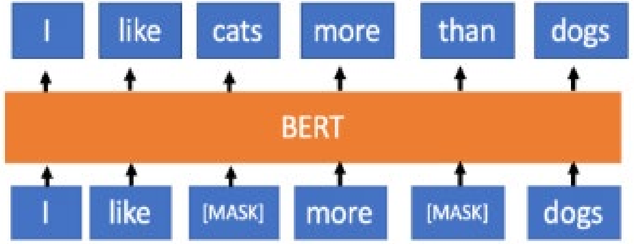
- Eliminate independence assumption with "*Permutation Language Modeling*"
  - Basically, consider predictions of multiple permutations of words in a sequence
- Even more complex
  - The model learns multiple ways to predict each sequence given different parts of the context



above. Specifically, we train on 512 TPU v3 chips for 500K steps with an Adam weight decay optimizer, linear learning rate decay, and a batch size of 8192, which takes about 5.5 days. It was

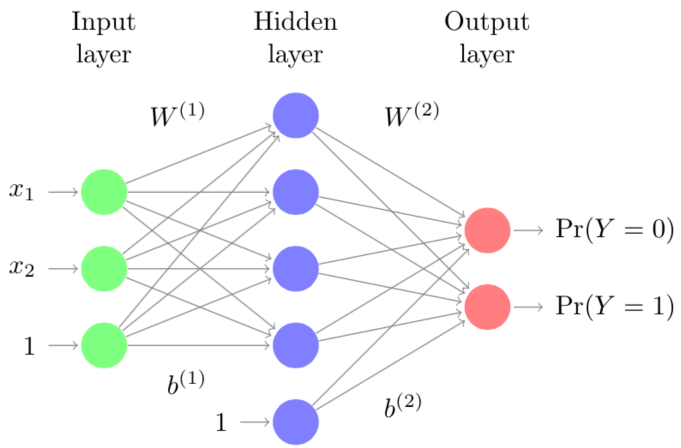
(that's $160k to train)

(in contrast, Bert used 64 TPUs for 4 days for a "mere" $14k)



Multiple Models in NLU Pipeline

- **Intent Classification** is often performed with SVM or FastText models
  - **Use Multi-class Support Vector Machine** to decide amongst multiple intents
    - tf-idf, n-gram frequency, embeddings, all potential features for SVM
    - Binary classifier for every pair of intents
      - "is account_balance" vs. "is open_credit", etc.
      - Simple vote: increase an intent's class count by 1 each time it wins one of the binary classifiers; take the highest as the intent label
    - **Advantage:** SVM is fast to train and infer; accuracy > 90% on standard workloads
  - **FastText** used to classify sequences into "topics"
    - Just create "topics" to be intents
    - Model takes sequences of words as inputs, embeds them, trained to select among multiple classes
    - **Advantage:** Fast (with pretrained embeddings); more accurate
    - Robust against misspellings
      - Words embedded in 3-character sequences:
        Kevin becomes: <Ke, Kev, evi, vin, in>



Stateful Classification

- Clinc
  - Each state associated with a separate intent classifier
    - Create an SVM/FastText model with each outgoing edge as a possible intent class
    - **Advantage:** State makes it easier to discern between intents
      - There are typically fewer intent classes to choose from in a given state
- DialogFlow
  - Coerce model outputs using Contexts
    - Classification model probabilities are changed based on Context
      - e.g., “2x more likely” to choose intent A over B in context C
    - **Advantage:** Reduced overall training (there’s no per-state classifier to train)
- Rasa
  - Touted as “stateless”
    - You give it training data that captures state (e.g., which intents should come next)
    - **Advantage:** Purely example based. Rasa scales well as a resule

Slot Extraction

- Can be thought of as a Sequence-to-Sequence task
  - Turn an utterance into an IOB representation
    - Yo fam get me a burger.
    - O B:person O B:person O B:food
  - Embed words, train a model to learn how to predict I, O, or B for each token

- Clinc
  - Per-competency slot extraction
    - Currently using Glove embeddings (olde but fast)
    - Bert embeddings improve accuracy (but radically increase training time)
- DialogFlow and Rasa
  - Appears per-intent, although model details are not immediately obvious

Summary

- We talked about Bert
  - Transformer, Attention, and WordPiece
- We talked a bit more about models under the hood
  - SVM/FastText for robust intent classification
  - Bert-like DNN for slot filling
- Next class: Ethics in NLP
  - Why is Google biased?

An example of Google bombing in 2006 that caused the search query "miserable failure" to be associated with George W. Bush and Michael Moore.