Carnegie Mellon University
Language
Technologies
Institute

11-411/11-611 Natural Language Processing

Document Classification

David R. Mortensen

September 22, 2022

Language Technologies Institute

Learning Objectives

At the end of this lecture, students will be able to:

- Distinguish document classification tasks from other NLP tasks.
- Name three or more document classification tasks.
- Explain how NB and logistic regression classifiers can be applied to each of these tasks.
- Understand, at a high level, how neural methods can be applied to the same tasks

A Step Back: NLP Tasks

NLP Projects Usually Involve Idenfying the Relevant Class of Tasks

- There are a small number of TASKS that show up again and again in NLP
  - Classification
  - Sequence labeling
  - Clustering
  - Language modeling
  - Parsing
  - etc.,
- Most NLP problems can be recast as one of these tasks
- Important goal of this course: learn to identify what kind of task a problem is

Classification is Labeling Items as Members of a Finite Number of Classes

Inputs:

- Set of classes $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$
- Set of items $x = \{x_1, x_2, \ldots, x_m\}$. These items can be anything, but in NLP they are frequently documents (the subject of this lecture).

Outputs:

- A set of tuples $L = \{\langle x_i, \ell_j \rangle_1, \ldots\}$ associating items with labels, for example $= \langle$"That's it. That's the tweet.", $\texttt{eng}\rangle$ labeling a tweet as English.

Classification requires a labeled training corpus.

Sequence Labeling is Classification of Items in Sequence

Sequence labeling is a special type of classification where items are classified based on their position in a sequence. For example, a sentence is a sequence of words and classifying the parts of speech of a sentence is a sequence labeling task.

Inputs:

- Set of classes $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$
- Sequence of items $x = [x_1, x_2, \ldots, x_t]$

Outputs:

- Sequence of labels $L = [\ell_1, \ell_2, \ldots, \ell_t]$ such that ℓ_1 is the label for x_1, ℓ_2 is the label for x_2, and so on.

Sequence labeling usually requires a labeled training corpus.

Clustering is Unsupervised Grouping of Items into Groups

Inputs:

- A set of items $x = \{x_1, x_2, \ldots, x_n\}$

Outputs:

- A partition of x into a set of k classes (where k may either be a hyperparameter or may be determined empirically)

Clustering requires no training data.

Language Model is Estimating the Probability of Sequences

Inputs:

- Sequence of items $x = [x_1, x_2, \ldots, x_{t-1}]$

Outputs:

- An estimate of the probability of the sequence or, equivalently, a probability distribution over x_t, $p(x_t|x_1, x_2, \ldots, x_{t-1})$

Language modeling does not require a label training set, but it does require a large among of unlabeled training data.

Parsing is Uncovering Latent Structure

Inputs:

- Sequence of items $x = [x_1, x_2, \ldots, x_n]$

Outputs:

- A graph G or other structured representation of structure latent in x. For example, G could be a (syntactic) dependency graph corresponding to x.

Parsing can be rule-based, unsupervised, or supervised.

Often Tasks Overlap

- A single model make include aspects of more than one task
- For example, HMMs (Module 5) are, in some sense, both sequence labeling models and language models
- Many complex problems can be divided into multiple tasks
    - Document classification can often be treated as a language modeling task (using large language models like BERT) and a classification task (using a simple classifier like a feed-forward neural network)
    - Dependency parsing (see example above) often involves classification (Module 7).
    - Etc.,

Document Classification as a General Task

One Important Classification Task is Document Classification

One of the most frequent, simple, NLP tasks is document classification.

Inputs:

- Set of classes $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$
- Set of documents $x = \{x_1, x_2, \ldots, x_n\}$

Outputs:

- A set of tuples $L = \{\langle x_i, \ell_j \rangle_1, \ldots\}$ associating items with labels, for example $= \langle \text{“}\mathrm{thanks\ I\ hate\ it.”}, -1 \rangle$ labeling the tweet as negative in sentiment

Some Document Classification Tasks

- Spam detection (Is this email message spam?)
- Subject identification (What is the topic of this article?)
- Language identification (What language is this web page written in?)
- Authorship attribution (Who wrote the Federalist Papers?)
- Toxic speech detection (Did this Facebook user just do a racism?)
- Sentiment analysis (Is this review positive or negative?)

Sentiment Analysis

Sentiment Analysis is a Useful Document Classification Task

Sentiment analysis is classifying documents according to the author's attitude towards the topic.

Polarity and Intensity

- Sentiment is sometimes characterized along two dimensions:
  - Polarity (whether the sentiment is positive or negative)
  - Intensity (whether the sentiment is strong or weak)
- Some SA datasets and tasks may classify documents in primarily one of these dimensions
- For example, star ratings on Amazon product reviews mostly indicate **polarity**
- Of course, three-star ratings may indicate low **intensity** as will us ambivalent polarity

Datasets

Amazon Product Data 142.8 million Amazon product reviews

Stanford Sentiment Treebank 10,000 Rotten Tomatoes movie reviews with polarity rated from 1 to 25.

IMDB Movie Reviews Dataset 25,000 user reviews from IMDB with binary ratings.

Sentiment140 Twitter user responses to products, brands, and topics rated of five-point scale.

Twitter US Airline Sentiment Tweets of user experience with US airlines labeled positive, negative, and neutral

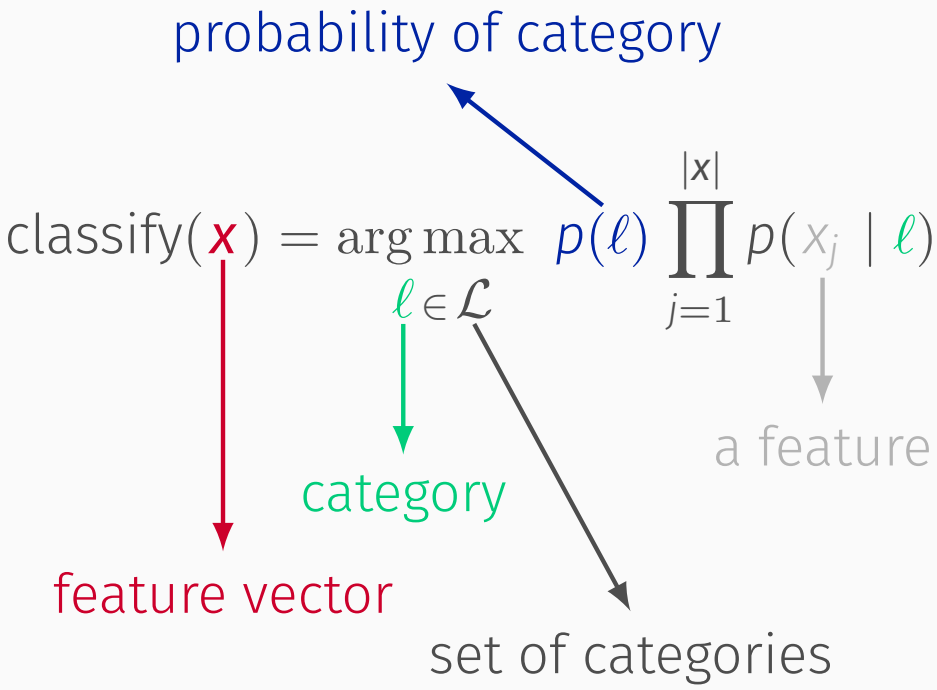
Paper Reviews Data Set 405 paper reviews from conferences rated on a five-point scale

Evaluation

- $\text{accuracy} = \frac{tp+tn}{tp+tn+fp+fn}$
- $\text{precision} = \frac{tp}{tp+fp}$
- $\text{recall} = \frac{tp}{tp+fn}$
- $F_1 = 2\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$

Review: Naive Bayes

- The feature vector $\boldsymbol{x}$ consists of the features that have been extracted from the document
- Each element of this vector, x_j, corresponds to a feature such as the presence of a particular word or ngram, the length of the document, and so on.
- There is a set of categories or labels, $\mathcal{L}$ (like 1 star, 2 stars,..., 5 stars).
- Each of these labels, ℓ, corresponds to a category like "five stars" or "negative polarity."
- $p(\ell)$ is the "base" probability of ℓ, disregarding any of the features in $\boldsymbol{x}$



But what about $p(x_j \mid \ell)$? How do we calculate it?

Three kinds of Naive Bayes

How you calculate $p(x_j \mid \ell)$ varies depending on what kind of classifier you are developing. There are three of these:

- **Gaussian Naïve Bayes** Considers a feature vector of continuous variables
- **Multinomial Naïve Bayes** Considers a feature vector representing frequencies
- **Bernoulli Naïve Bayes** Considers a vector of binary features

For sentiment analysis, and other document classification tasks, Multinomial Naïve Bayes is generally best.

Training NB Made Simple

- **For bag of words alone** (which is not always what we want for sentiment analysis) we can simplify things
- Put all of the documents with category ℓ into one "super document" (which we will call ℓ)
- Assume w_i is a word in V (the vocabulary of all words in the corpus)

$$\hat{P}(w_i \mid \ell) = \frac{\text{count}(w_i, \ell)}{\sum_{w \in V} \text{count}(w, \ell)}$$

- $\hat{P}(w_i \mid \ell)$ is the number of times that w_i occurs in ℓ over the sum of all words occurring in ℓ
- This is easy to calculate

Something is Rotten in the Kingdom of Denmark

What happens if $count(w_i, \ell) = 0$? Since the likelihoods of the features are simply multiplied together, zero for one feature means disaster. As with language modeling, we address this with smoothing. But unlike language modeling, Laplace (add one) smoothing is often good enough:

$$\hat{P}(w_i \mid \ell) = \frac{count(w_i, \ell) + 1}{\sum_{w \in V}(w, \ell + 1)} = \frac{count(w_i, \ell) + 1}{(\sum_{w \in V} count(w, \ell)) + \mid V \mid}$$

That's what we will do for sentiment analysis.

Features and Feature Extraction

- Sometimes, bag-of-words words are good enough for sentiment analysis
- However, bag-of-words vectors are very sparse and most of the features are not informative
  - Some uninformative features can be eliminated by taking out stop words
  - The classifier can also learn which features are useful
  - It is common to use **sentiment dictionaries** to bias the set of features in favor of words correlated with sentiment
- Furthermore, there are features one might like to include that are not words, or even counts (e.g. ngrams)

Logistic Regression for Sentiment Analysis

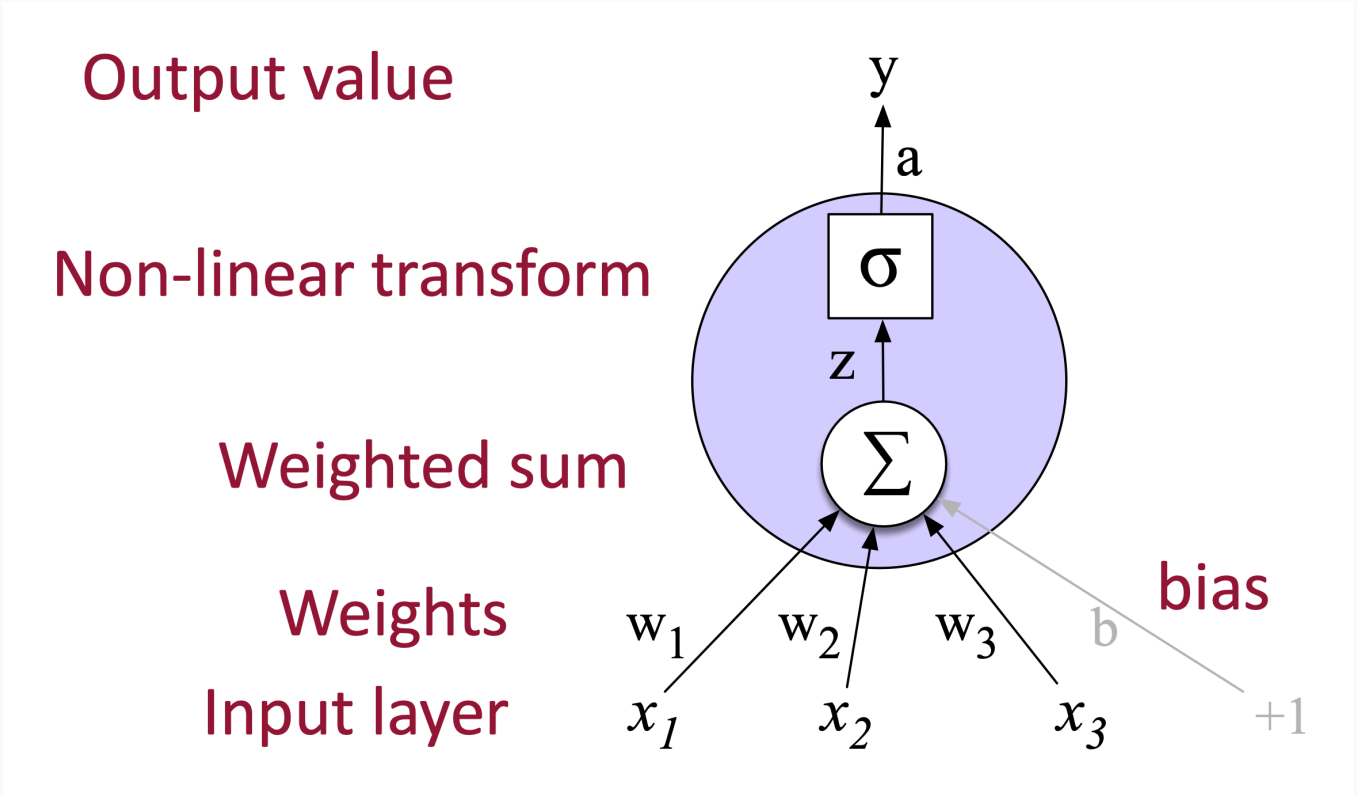
- Logistic regression works very well for sentiment analysis too
- It can work with or without a lot of FEATURE ENGINEERING
- Feature engineering is optimizing the set of features for a specific USE CASE
- LR learns to weight features according to how discriminative they are (and ignore meaningless features, so feature engineering should be less necessary
- However, in practice, dense, engineered feature vectors often work better for (LR as well as NB)

Neural Document Classification

We will now introduce contemporary approaches to document classification—especially sentiment analysis. We will talk about neural networks and neural language models, which we have not yet introduced. Don't worry if you don't understand everything. In Module 3, the details will become clear.

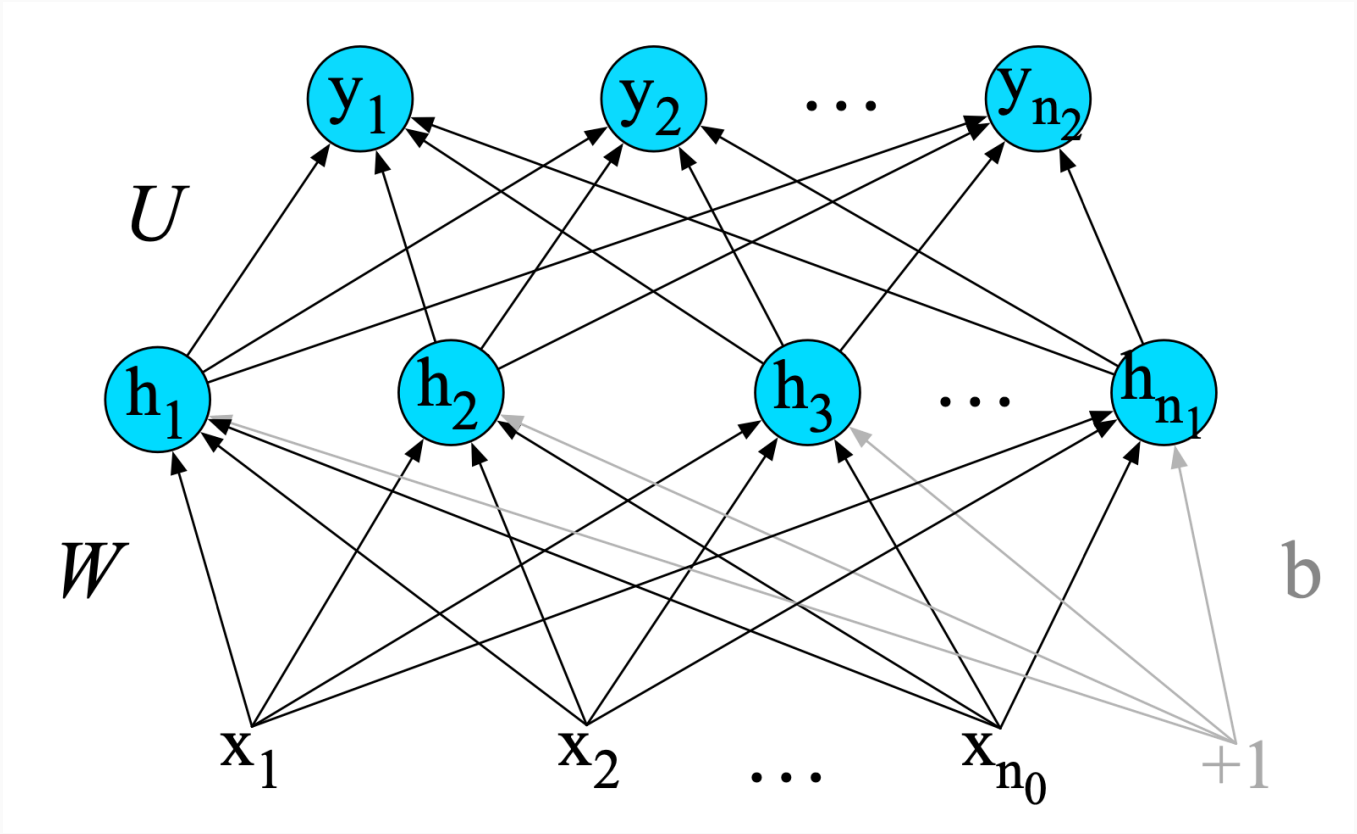
A Unit of a Neural Network

The components of a unit of a NN should be familiar from the LR lecture:



$$y = \sigma(w \cdot x + b) \tag{1}$$

Feedforward Neural Nets Have Many Units in Multiple Layers



Softmax is a Generalization of Sigmoid

Softmax will show up multiple times in this class as a way of converting numbers into probablities. For a vector z of dimensionality k, the softmax is:

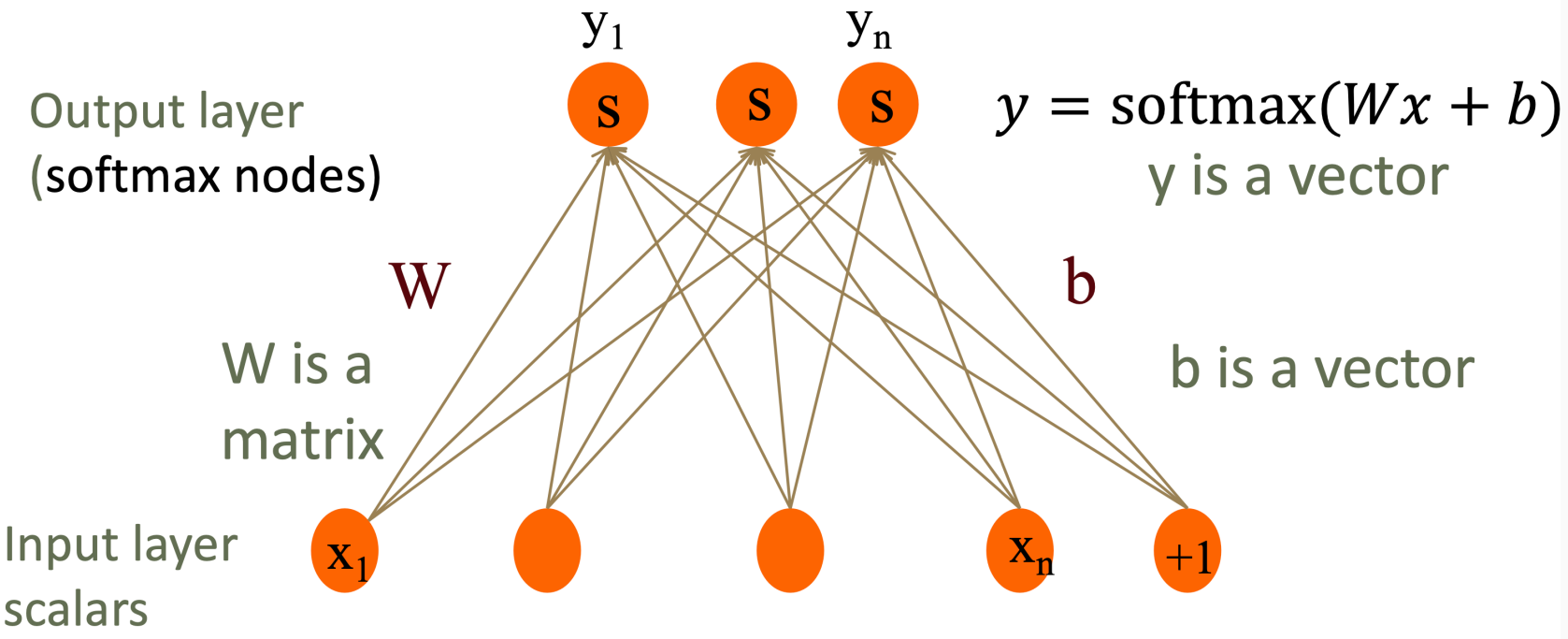
$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k} exp(z_i)}, \ldots, \frac{\exp(z_n)}{\sum_{i=1}^{k} exp(z_i)} \right] \tag{2}$$

$$\text{softmax}(z_i) = \frac{\exp(z_1)}{\sum_{j=1}^{k} exp(z_j)} \; 1 \leq i \leq k \tag{3}$$

For example, if $z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$ then $\text{softmax}(x) = [0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$

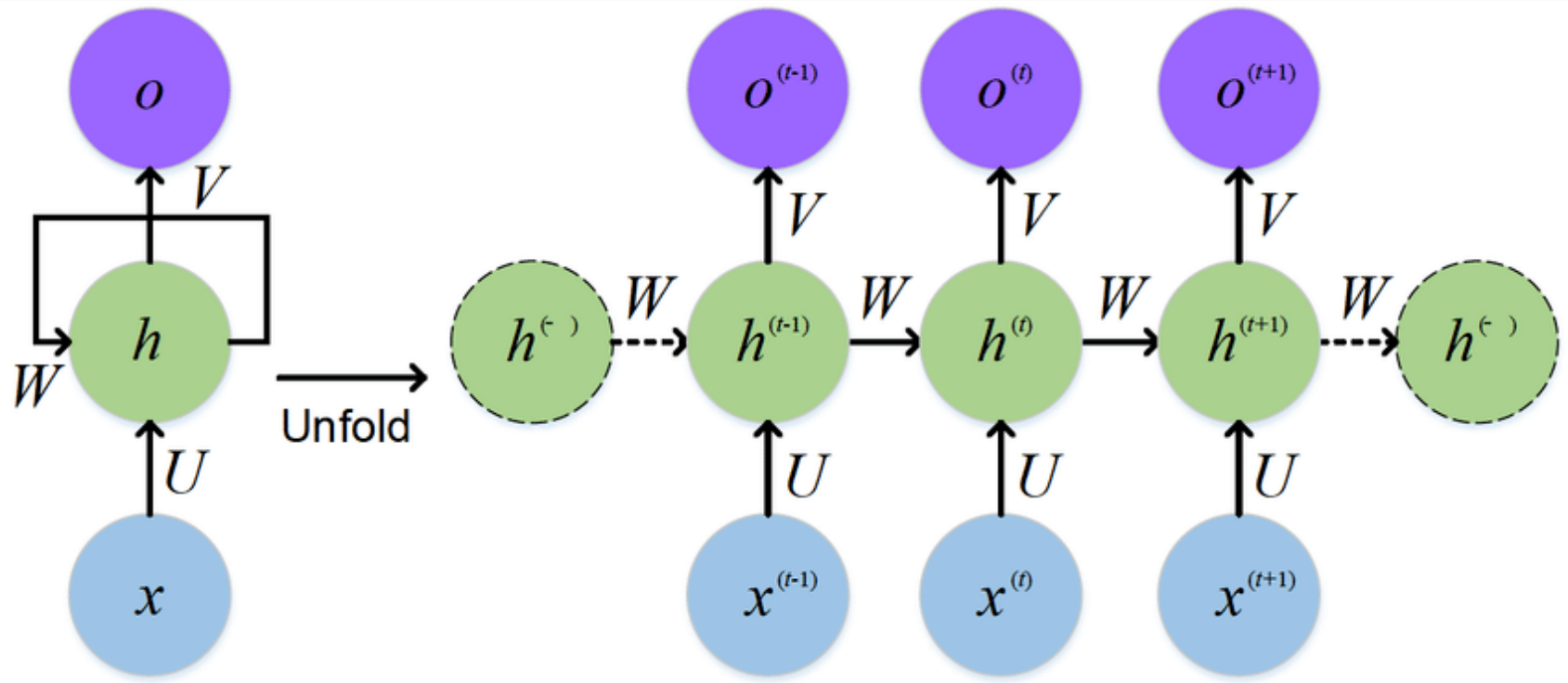
Multinomial Logistic Regression as a 1-layer Network

Fully connected single layer network



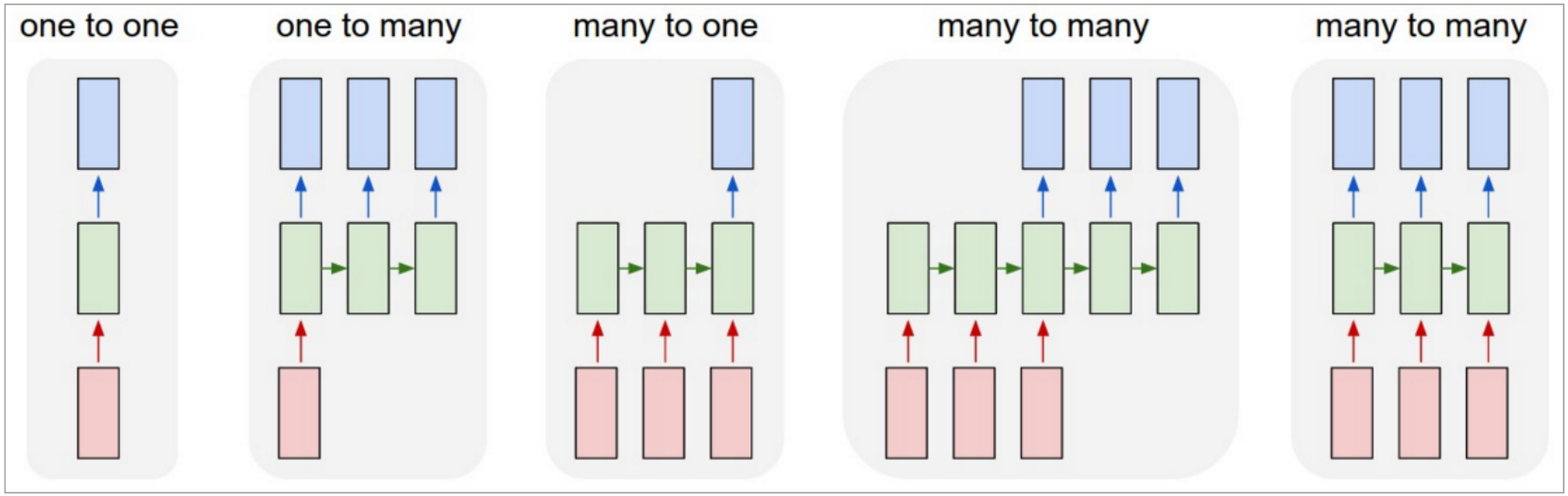
Foreshadowing: the Architecture of an RNN

As we will see in Module 5, RNNs are a special kind of multilayer neural network for modeling sequences. The “hidden” layers between the input and the output layer receive input not just from the input layer, but also from the hidden layer at a preceding timestep:



This means that RNNs can “remember” information from earlier in

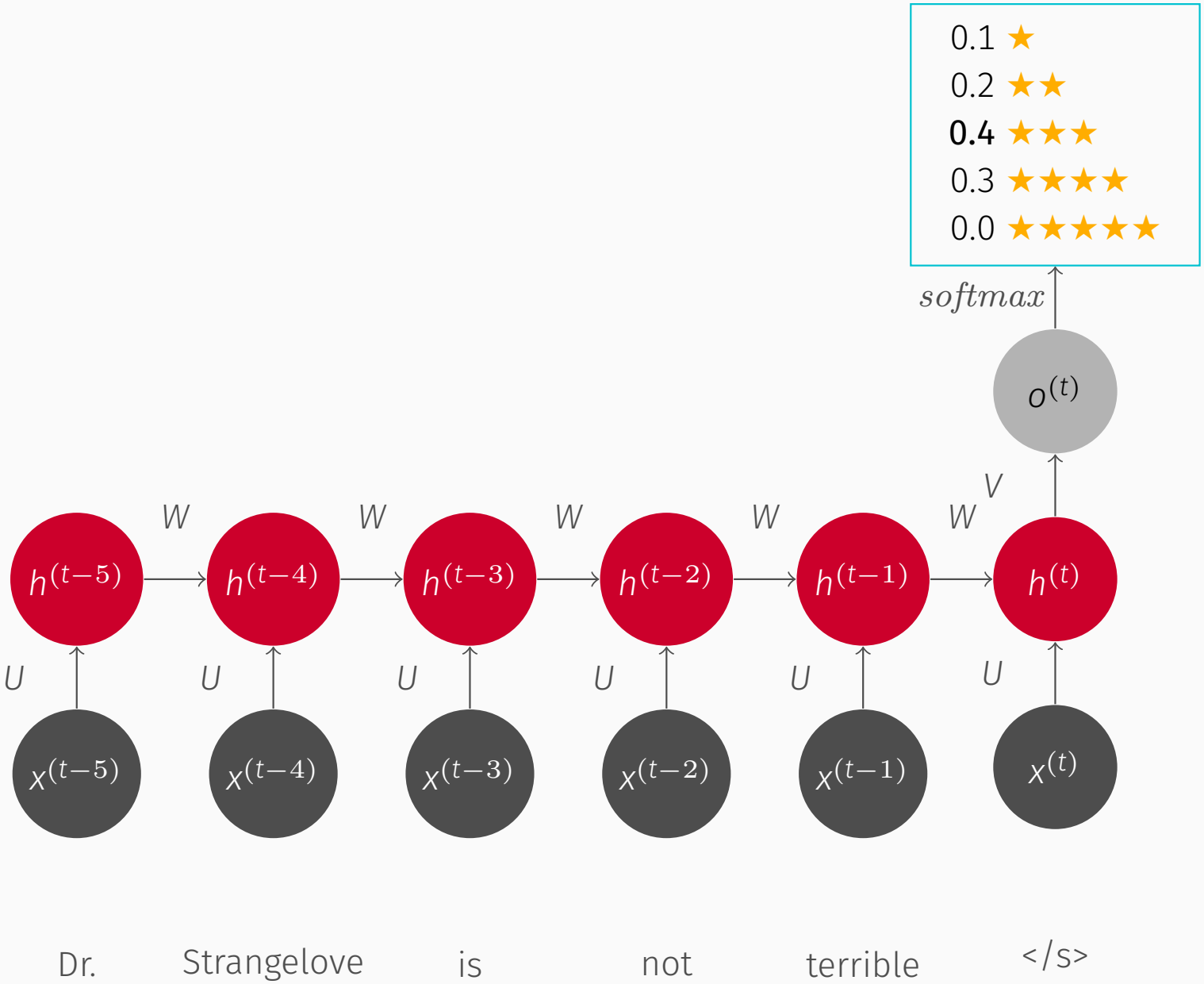
RNNs Can Be Used Various Ways, One of which is Classification



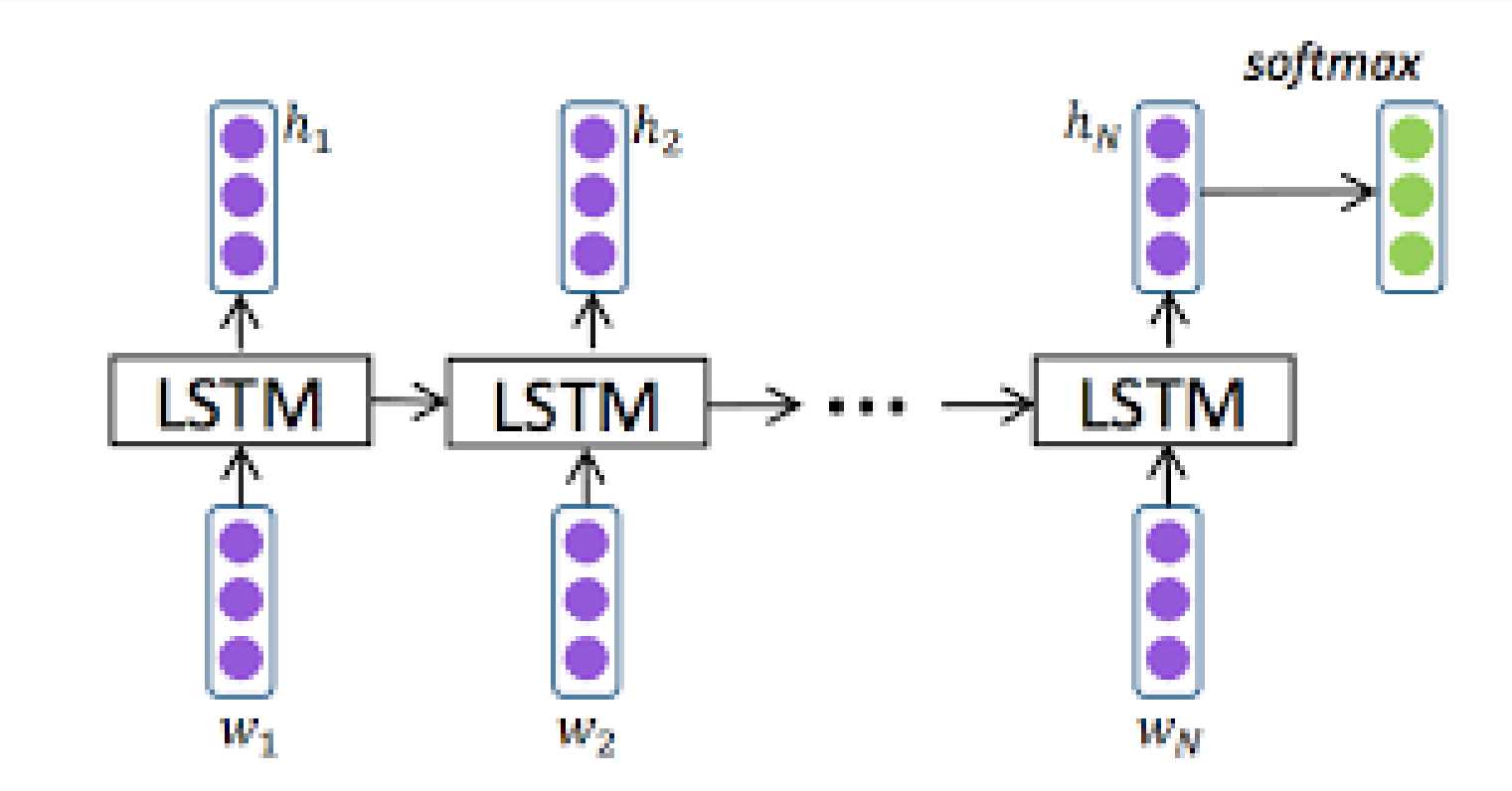
Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.

When an RNN classifies a sequence (e.g., a document) it is performing a many-to-one task (the third diagram).

Sentiment Classification with RNNs



Sentiment Classification with RNNs



Practical Concerns

What are the advantages of RNNs (especially LSTMs) over NB or LR for sentiment analysis?

- NB and LR classifiers have no sense of sequence (consider the bag-of-words representation of documents)
- RNNs are sensitive to sequence
- For NB and LR, we had to do special feature-engineering hacks to incorporate knowledge of negation
- This should not be necessary with RNNs
- In fact, feature engineering more or less goes away with RNNs
- However, RNNs are **much** less efficient than NB classifiers

Limitations of RNNs

- RNNs are not very efficient
- Traditional RNNs are not very good at dealing with long histories/sequences (LSTMs are better)
- There are now approaches that perform better

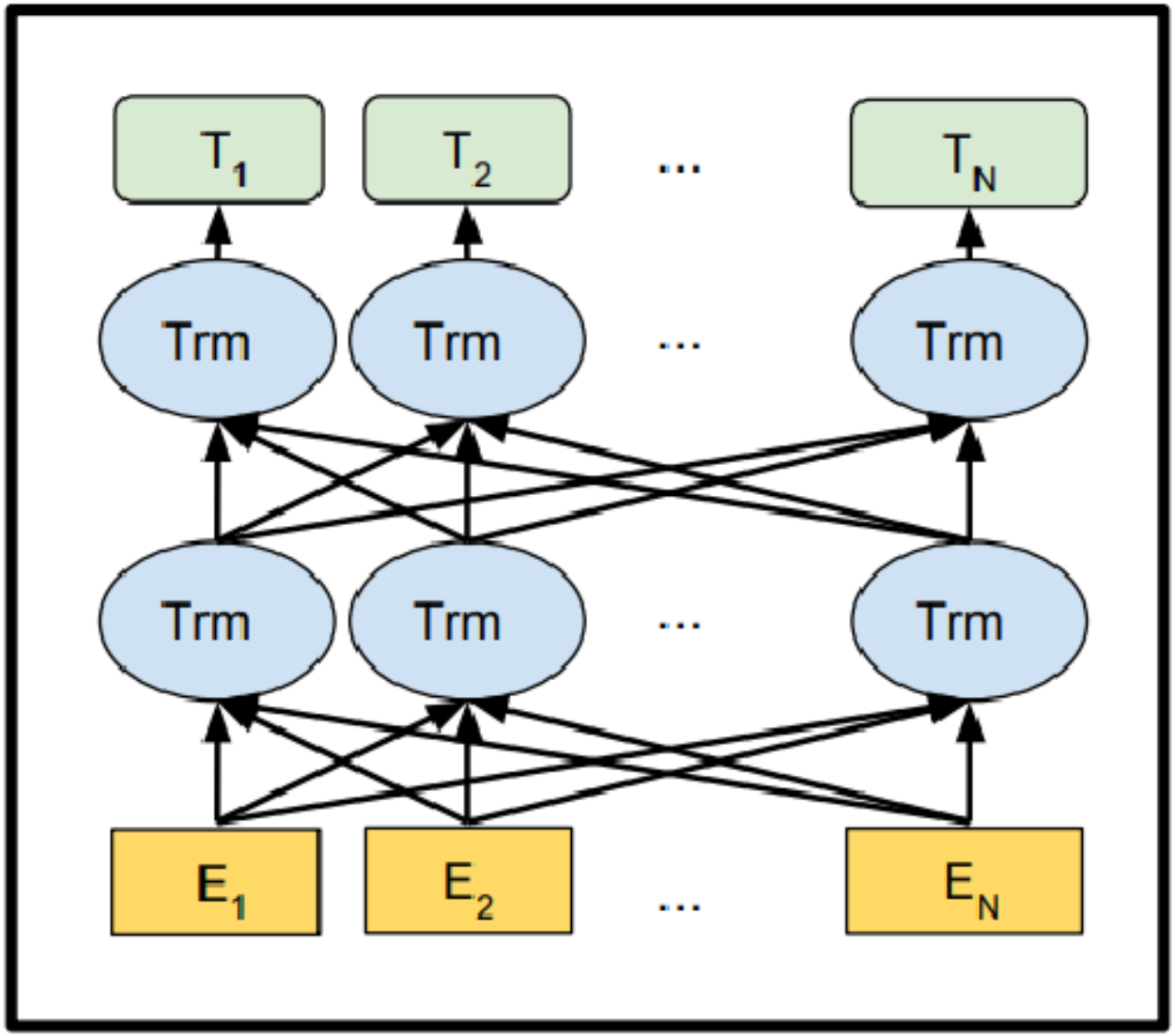
What is BERT?

- BERT is a large pretrained language model (a model the estimates the probability of a sequence of words for which you download all of the parameters rather than training them yourself)
- It is based on the transformer architecture
- It and its variants have been wildly successful at a range of NLP tasks
- We will talk more about it in Module 4.

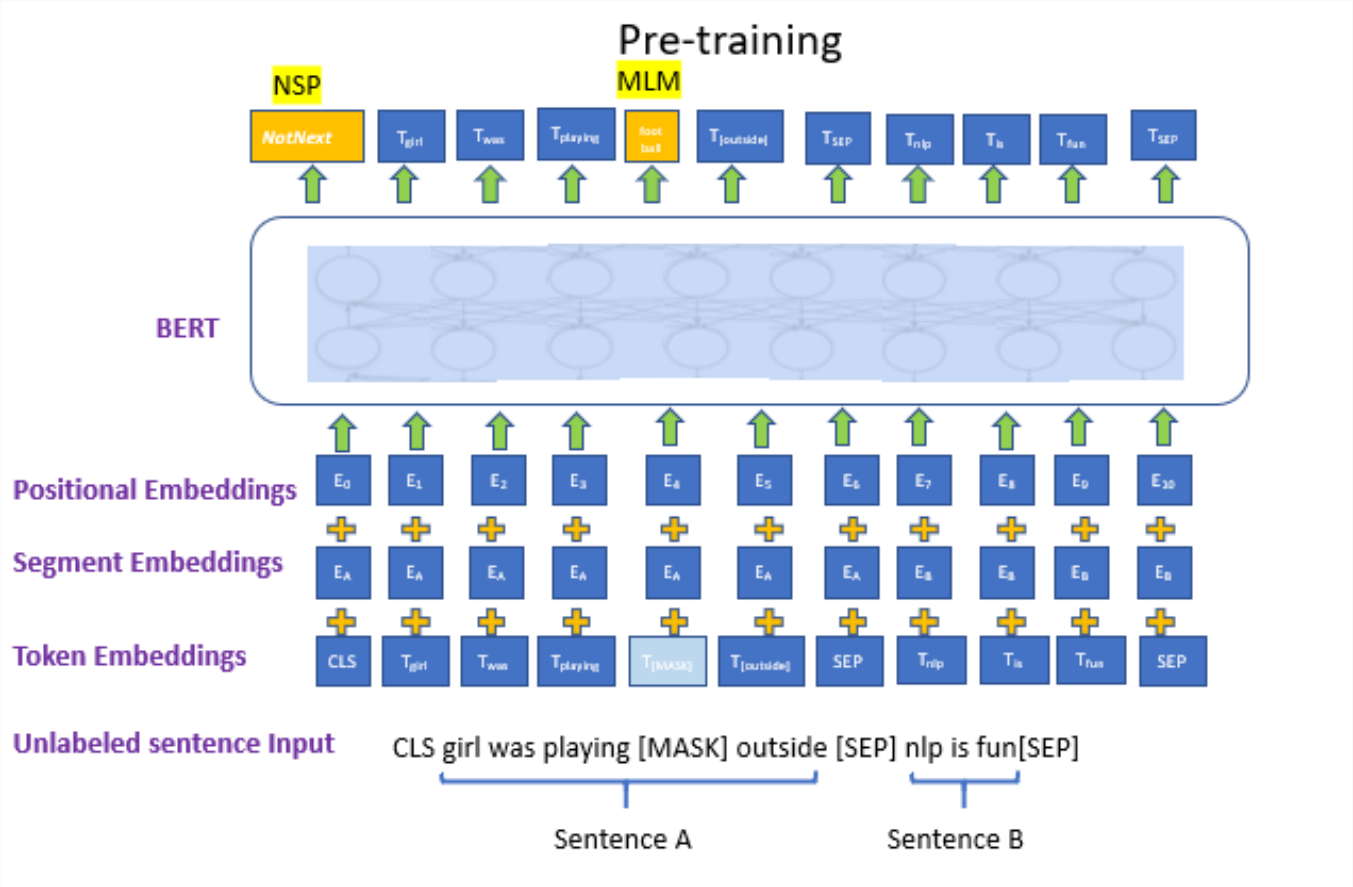
Overview: Advantages of BERT

- Pre-trained models easily tuned for different tasks, including classification tasks like sentiment analysis
- Produces contextual representations of words (not, e.g., one embedding per word)
- Scales well

The Architecture of BERT



The Pre-Training of BERT

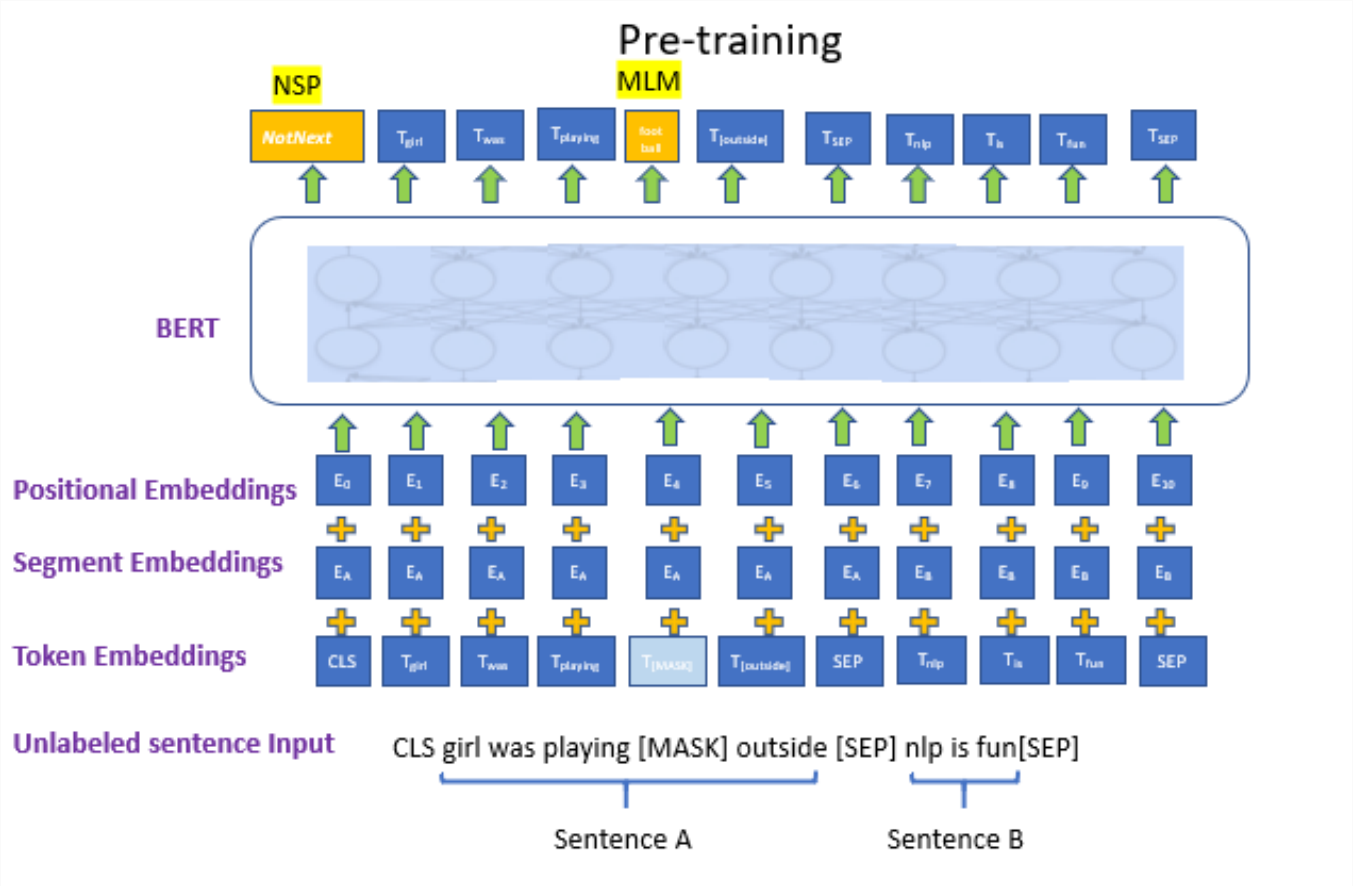


BERT learns *contextual* representations. Two training objectives:

MLM Masked Language Model.

NSP Next Sentence Prediction.

Some Tokens of Our Friendship

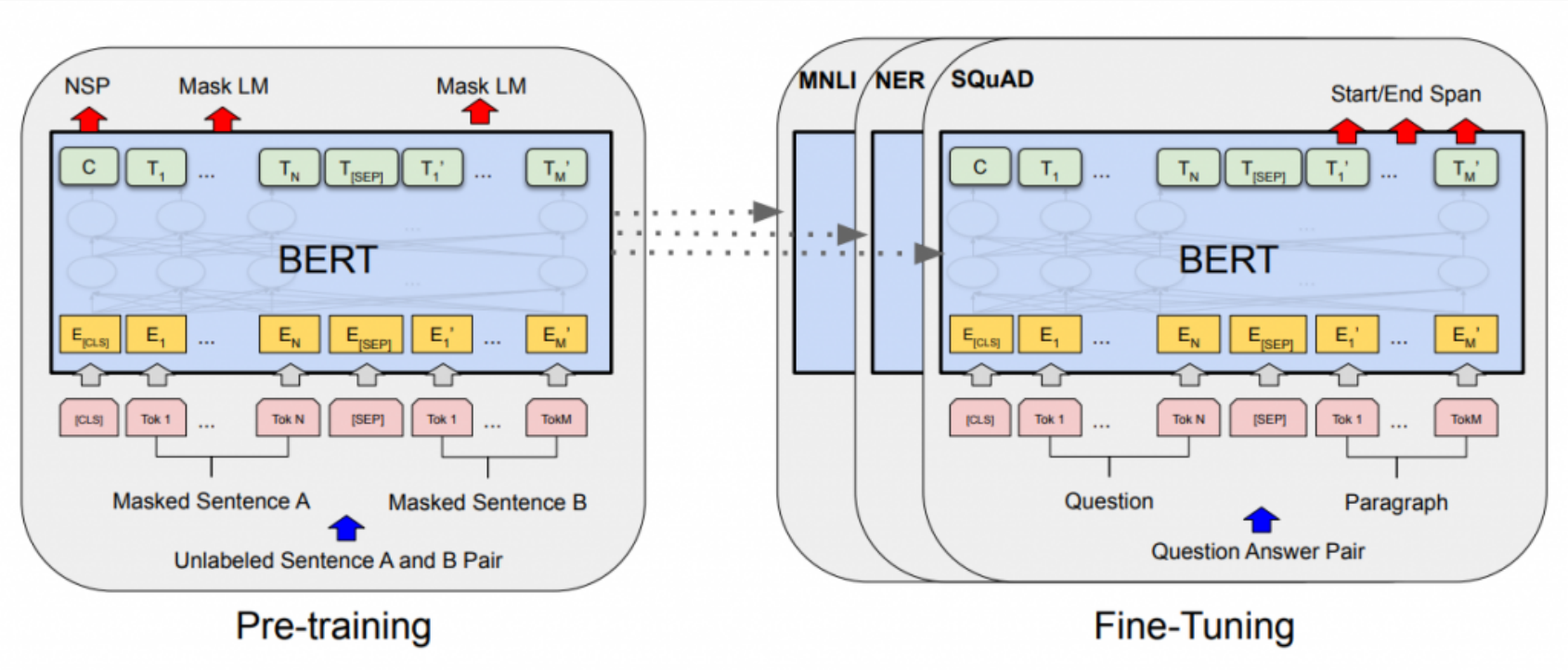


BERT has its own tokenizer, which you must use when you use BERT.
Special tokens:

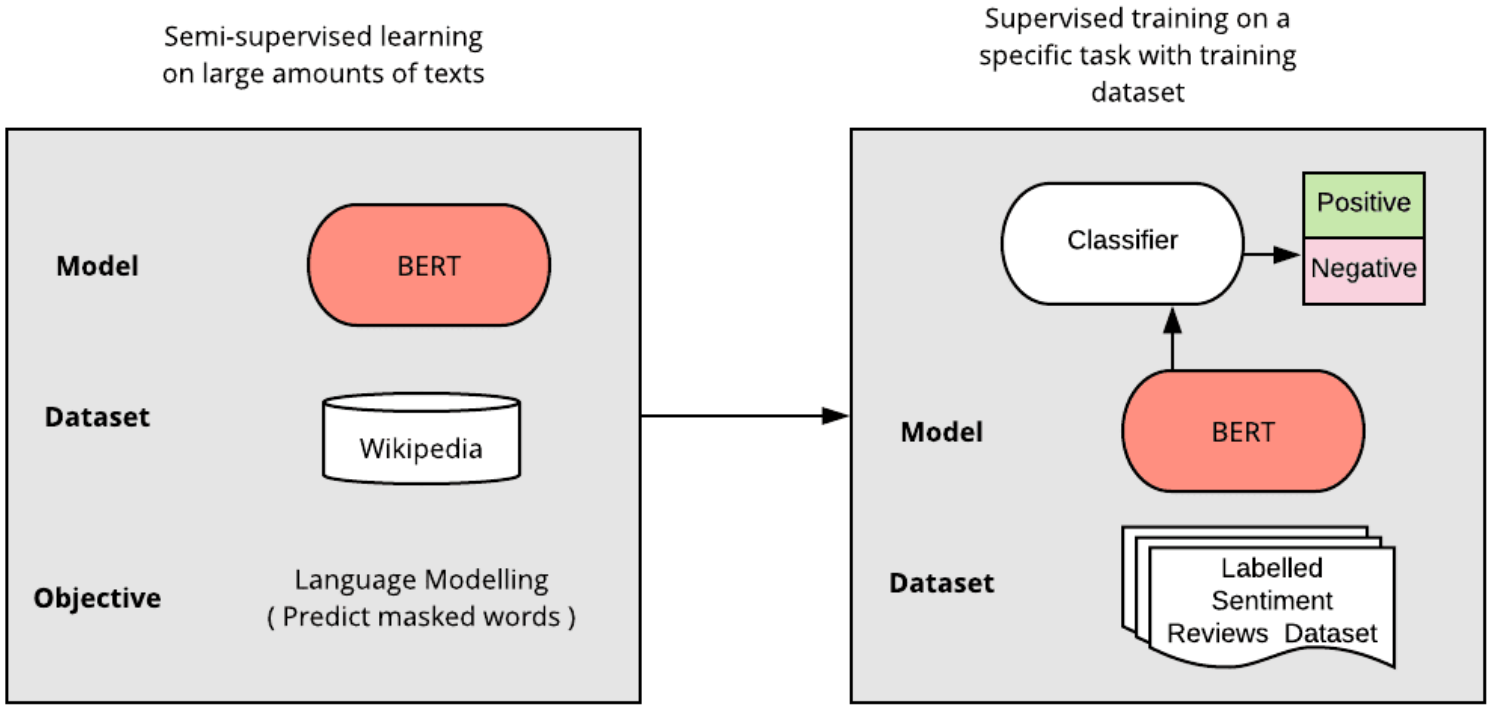
- [CLS] “This is a classification task” token
- [SEP] Separator token

Fine Tuning BERT

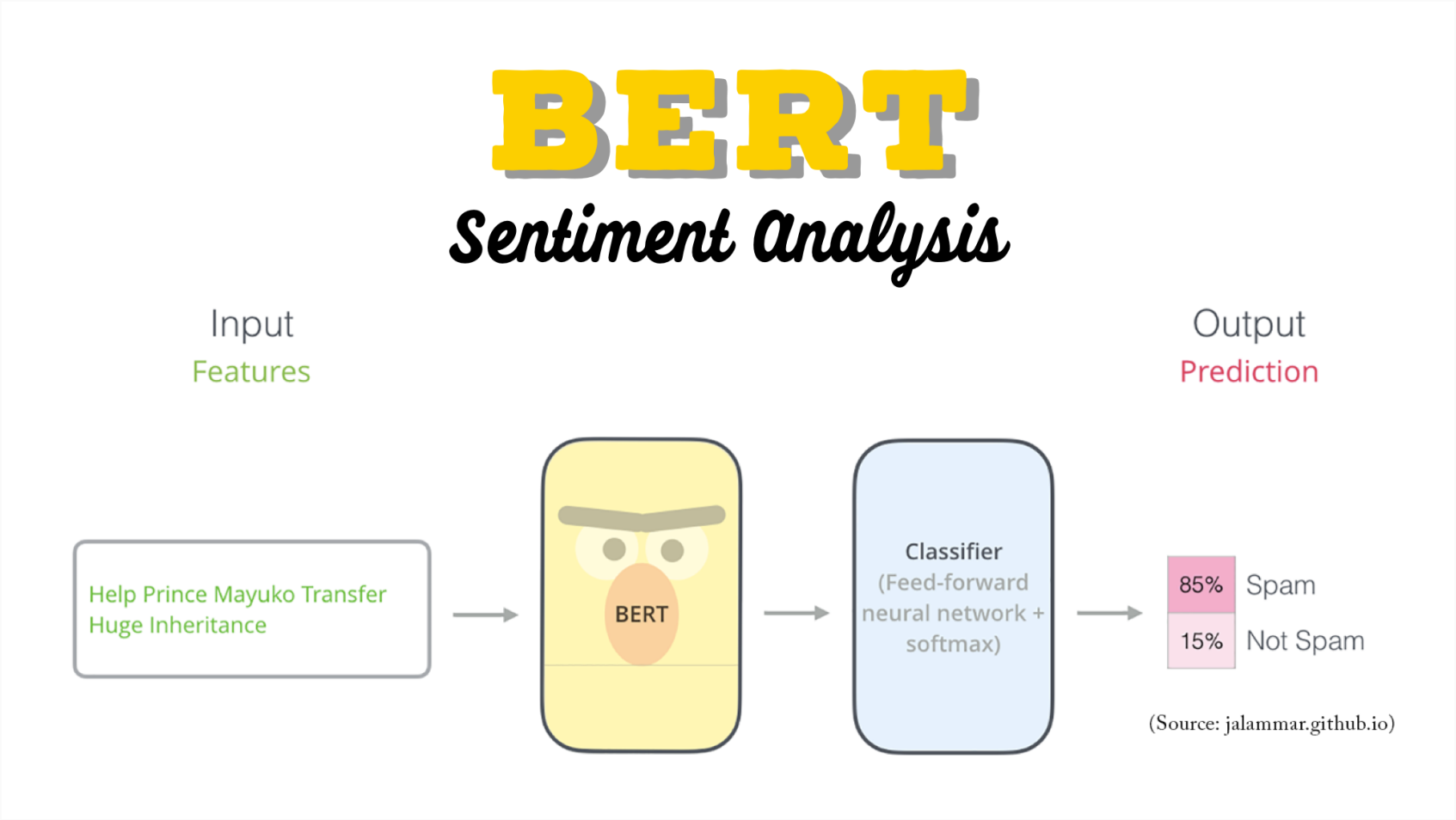
BERT is pretrained in a very generic fashion. Then it can be turned for any number of more specific tasks, including sentiment analysis.



Fine Tuning BERT



Using BERT in a Document Classification Task



Practical Concerns

BERT has been very effective in a range of tasks. It has been the standard approach to sentiment analysis (at least for English) for some time. Advantages:

- Performance is much better than that of previous models
- Can handle large volumes of data efficiently
- One general model can easily be adapted to many specific tasks (**including various sentiment analysis datasets**); this takes some of the "black magic" out of text classification and other NLP tasks
- Multilingual BERTs exist (including mBERT) that extend BERT beyond English; these have been used for **multilingual sentiment analysis** and many other things
- If you are building a sentiment analyzer today and have a large dataset in English or another major language, you should try BERT first

Language ID

For Homework Assignment 3, You Will Implement Language ID

- You will implement a language ID model using NB
- Structurally, this task is very similar to sentiment classification
  - There are $|\mathcal{L}|$ discrete classes
  - There are $|D|$ documents
  - You must assign a class in $\ell \in \mathcal{L}$ to each document $d \in D$
- Ways LID is different
  - $|L| > 2$ (multiway classification)
  - Words are not good features (why?)
  - Character ngrams are better
- **Food for thought:** Would you adopt a different approach if you were using LR?

Questions?