Carnegie Mellon University
Language
Technologies
Institute

11-411/11-611 Natural Language Processing

Document Classification

David R. Mortensen

February 9, 2023

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
  - Structured prediction
  - 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

Is this document in Swahili?

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.", **eng**$\rangle$ labeling a tweet as English.

Classification requires a labeled training corpus.

What is the part of speech of each word in the sentence, “’Twas brillig and the slithy toves did gyre and gimble in the wabe?”

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.

If there were five parts of speech in this unlabeled corpus from an under-documented language, which word would belong to each?

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.

“Wreck a nice beach” or “Recognize speech?”

Language Modeling 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.

What is the hierarchical structure of this sentence?

Structured Prediction 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.

Structured prediction 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 is a classification task, but it is common to use REPRESENTATIONS from large language models like BERT as input to the classifiers (this lecture)
  - 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 $= \langle \text{“thanks I hate it.”}, 0 \rangle$ 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

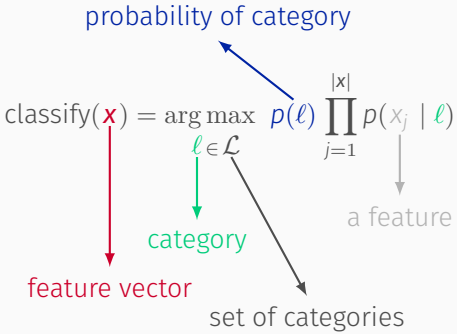
Paper Reviews Data Set 405 paper reviews from conferences rated on a five-point scale

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 (Homework 3)

- 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 $\boldsymbol{x}$



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[1]
  - The classifier can also learn which features are useful
  - It is common to use **sentiment dictionaries** to provide two highly informative features
- Furthermore, there are features one might like to include that are not words, or even counts (e.g. ngrams)

[1] Words frequent in most documents.

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 to ignore meaningless features $\Rightarrow$ less feature engineering)
- 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.

The Variables in Our Very Important Formula

- $\mathbf{x}$ A vector of features of n dimensions (like number of positive sentiment words, length of document, etc.)
- $\mathbf{w}$ A vector of weights of n dimensions specifying how discriminative each feature is
- b A scalar bias term that shifts z
- z The raw score
- y A random variable (e.g., $y = 1$ means positive sentiment and $y = 0$ means negative sentiment

The Fundamentals

The fundamental equation that describes a unit of a neural network should look very familiar:

$$z = b + \sum_i w_i x_i \tag{1}$$

Which we will represent as

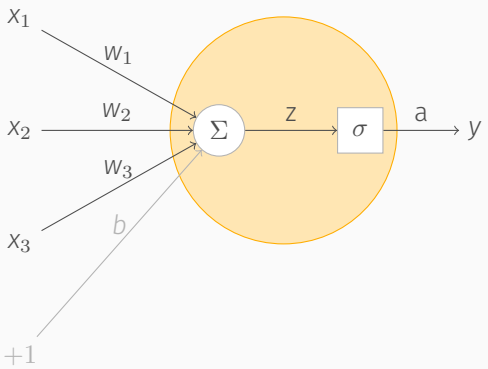
$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{2}$$

But we do not use z directly. Instead, we pass it through a non-linear function, like the sigmoid function:

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

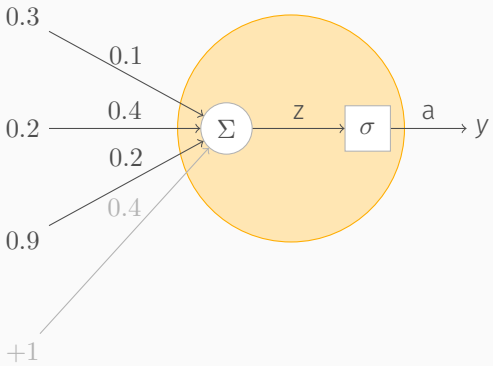
(which has some nice properties even though, in practice, we will prefer other functions like tanh and ReLU).

A Unit Illustrated

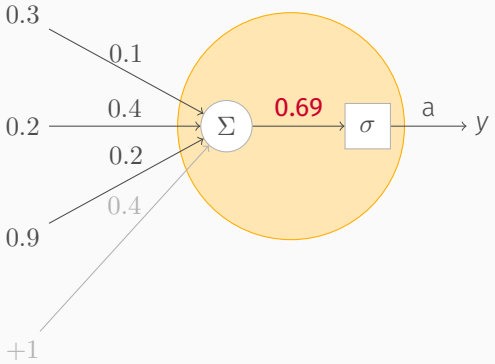


Take, for example, a scenario in which our unit has the weights [0.1, 0.4, 0.2] and the bias term 0.4 and the input vector *x* has the values [0.3, 0.2, 0.9].

Filling in the Input Values and Weights

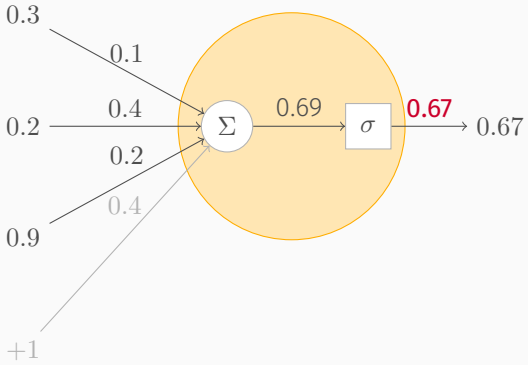


Multiplying the Input Values and Weights and Summing Them (with the Bias Term)



$$z = x_1 w_1 + x_2 w_2 + x_3 w_3 + b = 0.1(0.3) + 0.4(0.2) + 0.2(0.9) + 0.4 = 0.69 \quad (4)$$

Applying the Activation Function (Sigmoid)



$$y = \sigma(0.69) = \frac{1}{1 + e^{-0.69}} = 0.67 \tag{5}$$

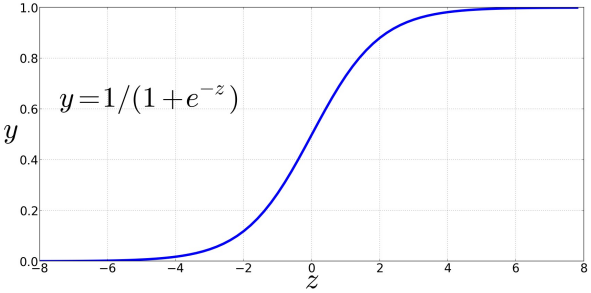
Non-Linear Activation Functions

Non-Linear Activation Functions

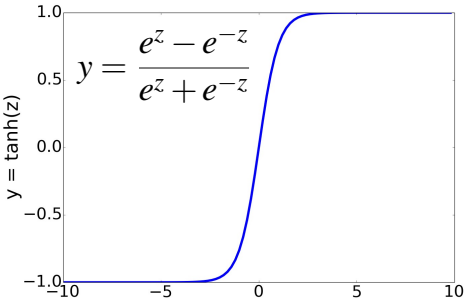
We're already seen the sigmoid for logistic regression:

Sigmoid

$$y = \sigma(z) = \frac{1}{1+e^{-z}}$$

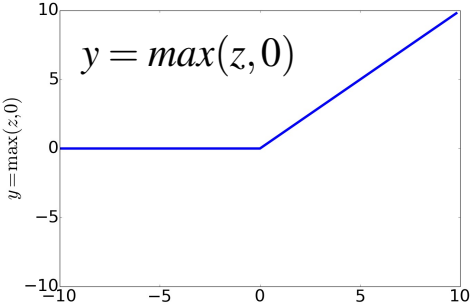


Non-Linear Activation Functions besides sigmoid



tanh

Most Common:



ReLU
Rectified Linear Unit

A Little Linear Algebra

So Far, We Have Assume You Know Dot Products

$$\boldsymbol{a} = (a_1, a_2, a_3)$$

$$\boldsymbol{b} = (b_1, b_2, b_3)$$

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

A matrix is an array of numbers

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

Two rows, three columns.

It’s Easy to Multiple a Matrix by a Scalar

$$2 \cdot \begin{bmatrix} 5 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 2 \cdot 5 & 2 \cdot 2 \\ 2 \cdot 3 & 2 \cdot 1 \end{bmatrix} = \begin{bmatrix} 10 & 7 \\ 2 & 4 \end{bmatrix}$$

Multiplying Matrices by Matrices Is Slightly Trickier

Let a_1 and a_2 be the row vectors of matrix A and b_1 and b_2 be the column vectors of a matrix B. Find $C = AB$

$$\begin{bmatrix} 1 & 7 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 3 & 3 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} a_1 \cdot b_2 & a_1 \cdot b_2 \\ a_2 \cdot b_1 & a_2 \cdot b_2 \end{bmatrix} = \begin{bmatrix} 38 & 17 \\ 26 & 14 \end{bmatrix}$$

A must have the same number of rows as *B* has columns.

Multiplying a Matrix by a Vector Is Roughly the Same

Multiplying a matrix by a vector is like multiply a matrix by a matrix with one column:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + ay + az \\ bx + by + bz \\ cx + cy + cz \end{bmatrix}$$

The result is a vector.

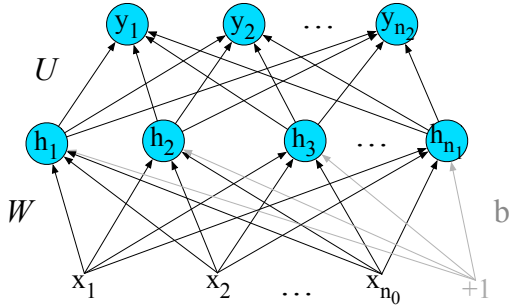
Matrix multiplication is not hard but inference with neural nets is mostly this (plus some non-linear functions)

Feedforward Neural Networks

Adding multiple units to a neural network increases its power to learn patterns in data. **Feedforward Neural Nets (FFNNs or MLPs)**

Feedforward Neural Networks

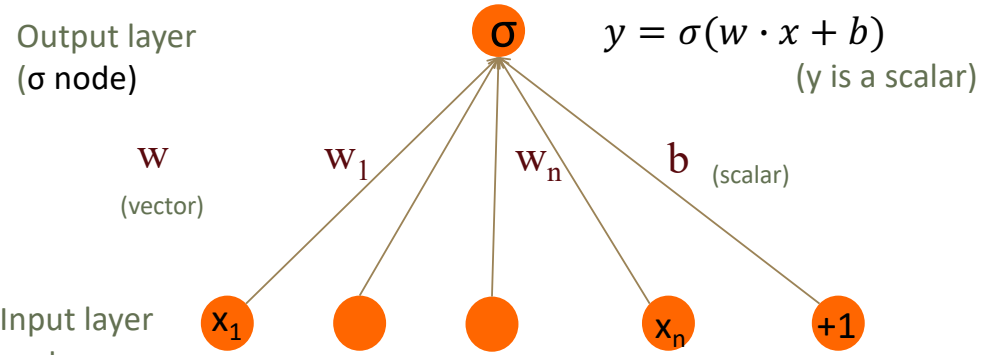
Can also be called **multi-layer perceptrons** (or **MLPs**) for historical reasons



The simplest FFNN is just binary logistic regression
(INPUT LAYER = feature vector)

Binary Logistic Regression as a 1-layer Network

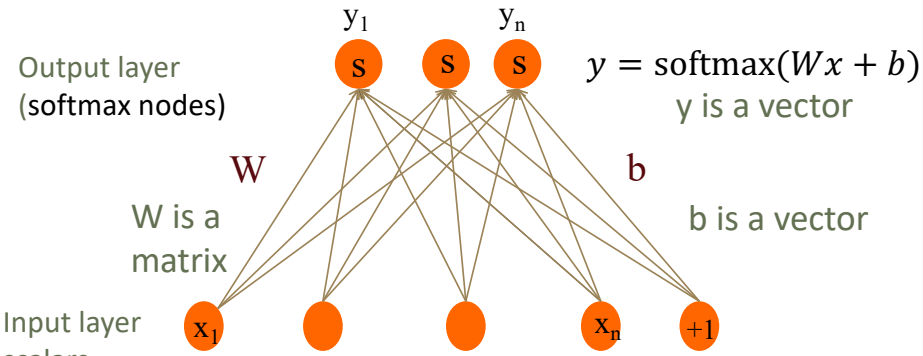
(we don't count the input layer in counting layers!)



Multinomial Logistic Regression is a slightly more complicated FFNN

Multinomial Logistic Regression as a 1-layer Network

Fully connected single layer network



x: feature vector; **W** is a matrix of weights; **b** is a vector of weights; **s**: softmax layer (see next slide); **y**: probabilities of categories

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:

$$\text{softmax}(\mathbf{z}) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k} exp(z_i)}, \ldots, \frac{\exp(z_n)}{\sum_{i=1}^{k} exp(z_i)} \right] \tag{6}$$

$$\text{softmax}(z_i) = \frac{\exp(z_1)}{\sum_{j=1}^{k} exp(z_j)} \; 1 \leq i \leq k \tag{7}$$

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

Probability distribution: a statistical function describing all the possible values/probabilities for a random variable within a given range.

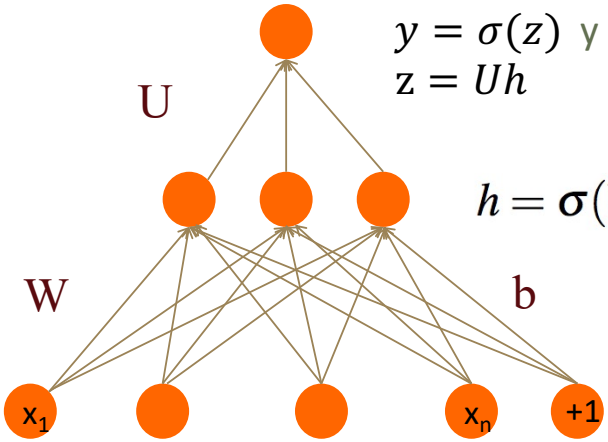
The real power comes when multiple layers are added

Two-Layer Network with scalar output

Output layer
(σ node)

hidden units
(σ node)

Input layer
(vector)



$y = \sigma(z)$ y is a scalar

$z = Uh$

$h = \sigma(Wx + b)$

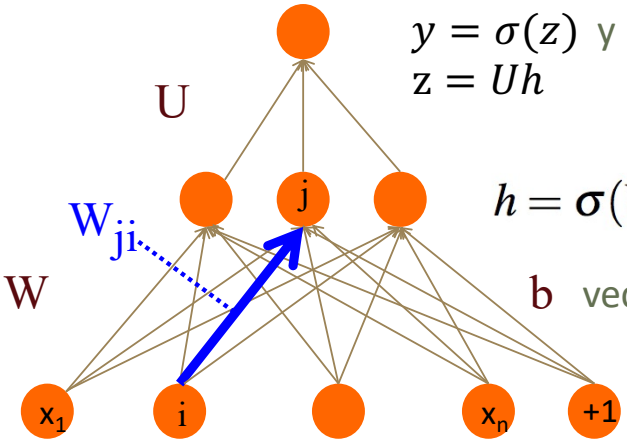
Could be ReLU
Or tanh

Two-Layer Network with scalar output

Output layer
(σ node)

hidden units
(σ node)

Input layer
(vector)



$y = \sigma(z)$ y is a scalar

$z = Uh$

$h = \sigma(Wx + b)$

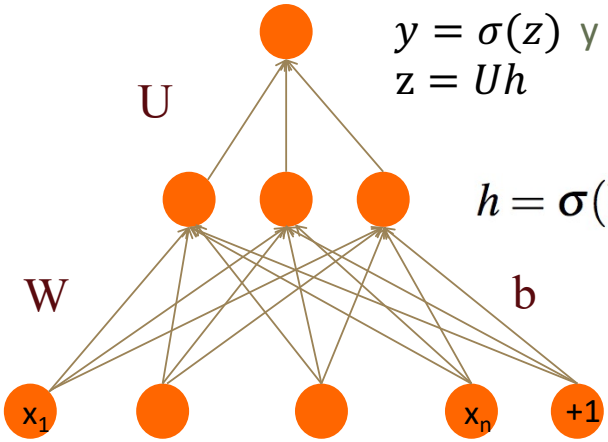
b vector

Two-Layer Network with scalar output

Output layer
(σ node)

hidden units
(σ node)

Input layer
(vector)



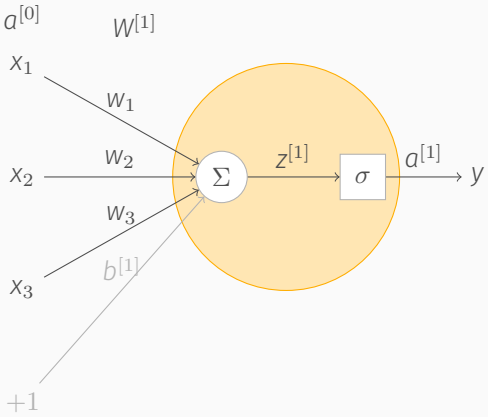
$y = \sigma(z)$ y is a scalar

$z = Uh$

$h = \sigma(Wx + b)$

Could be ReLU
Or tanh

A Forward Pass in Terms of Multi-Layer Notation



for each $i \in 1..n$ do

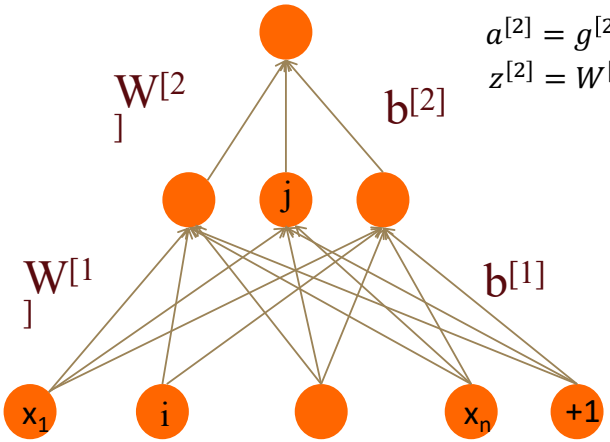
$z^{[i]} \leftarrow W^{[i]} a^{[i-1]} + b^{[i]}$

$a^{[i]} \leftarrow g^{[i]}(z^{[i]})$

end for

$\hat{y} \leftarrow a^{[n]}$

Multi-layer Notation



$y = a^{[2]}$

$a^{[2]} = g^{[2]}(z^{[2]})$ sigmoid or softmax

$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$

$a^{[1]} = g^{[1]}(z^{[1]})$ ReLU

$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$

$a^{[0]}$

Replacing the bias unit

Instead of:

$$x = x_1, x_2, \ldots, x_{n0}$$

$$h = \sigma(Wx + b)$$

$$h_j = \sigma\left(\sum_{i=1}^{n_0} W_{ji} x_i + b_j\right)$$

We'll do this:

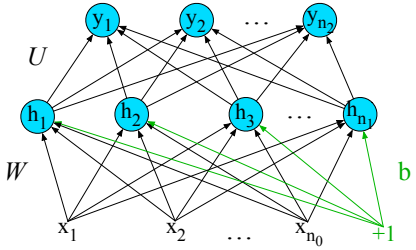
$$x = x_0, x_1, x_2, \ldots, x_{n0}$$

$$h = \sigma(Wx)$$

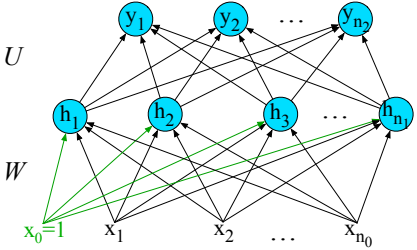
$$\sigma\left(\sum_{i=0}^{n_0} W_{ji} x_i\right)$$

Replacing the bias unit

Instead of:



We'll do this:



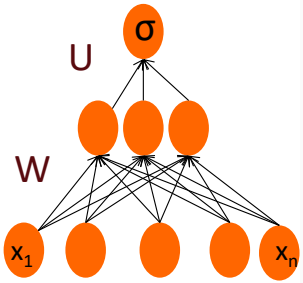
Feedforward Neural Nets as Classifiers

Classification: Sentiment Analysis

We could do exactly what we did with logistic regression

Input layer are binary features as before

Output layer is 0 or 1

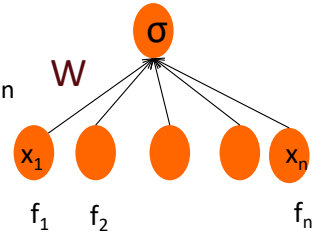


Sentiment Features

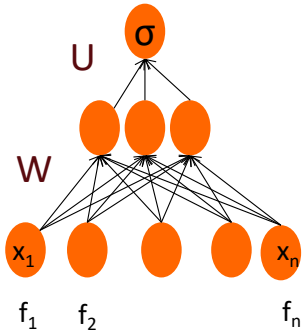
Var	Definition
x_1	count(positive lexicon) $\in$ doc)
x_2	count(negative lexicon) $\in$ doc)
x_3	$\begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_4	count(1st and 2nd pronouns $\in$ doc)
x_5	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$
x_6	log(word count of doc)

Feedforward nets for simple classification

Logistic Regression



2-layer feedforward network



Just adding a hidden layer to logistic regression

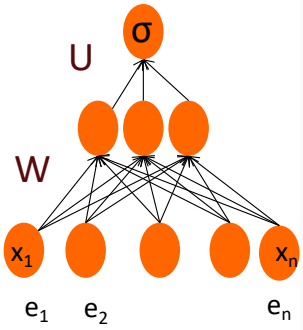
- allows the network to use non-linear interactions between features
- which may (or may not) improve performance.

Even better: representation learning

The real power of deep learning comes from the ability to **learn** features from the data

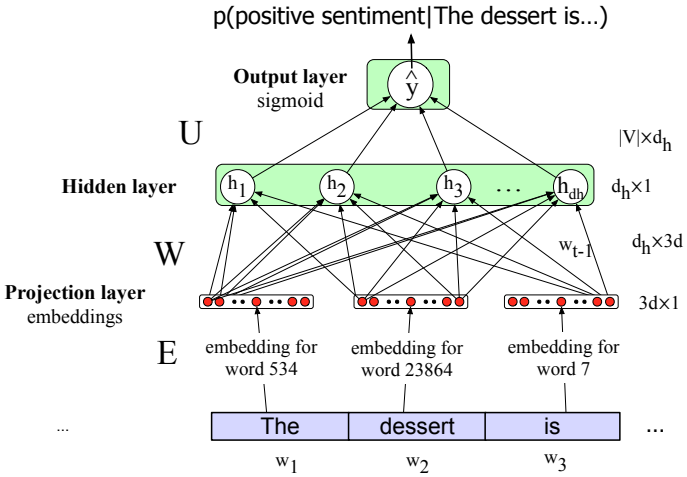
Instead of using hand-built human-engineered features for classification

Use learned representations like embeddings!



Embeddings are representations of items (e.g., words) as dense vectors

Neural Net Classification with embeddings as input features!



Documents Come in Different Sizes

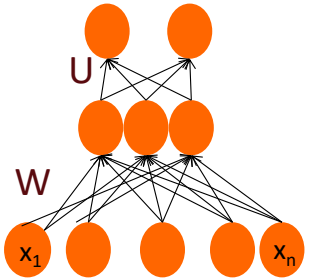
- Most documents are not three words long
- We have an embedding for each word
- We can pool the embeddings
  - Take the mean (mean pooling)
  - Take the max at each dimension (max pooling)
- Then we can use FFNNs with embeddings!

Reminder: Multiclass Outputs

What if you have more than two output classes?

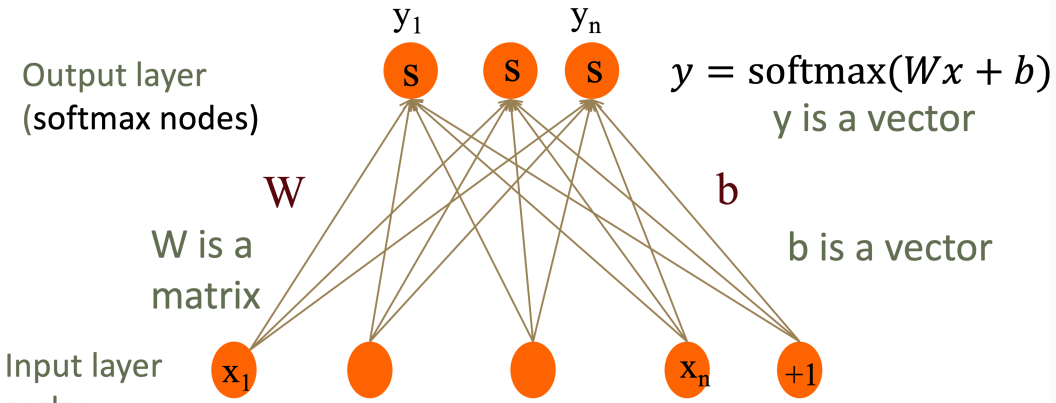
- Add more output units (one for each class)
- And use a “softmax layer”

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq D$$



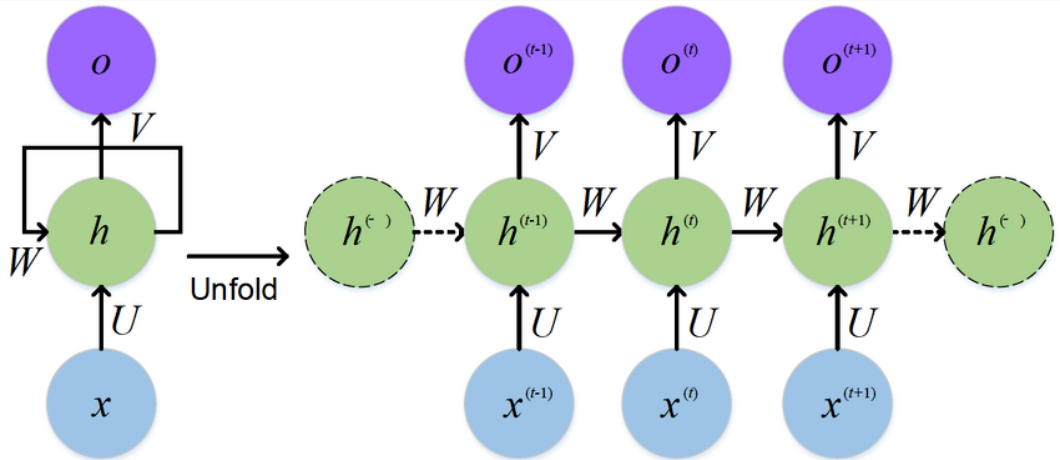
Multinomial Logistic Regression as a 1-layer Network

Fully connected single layer network

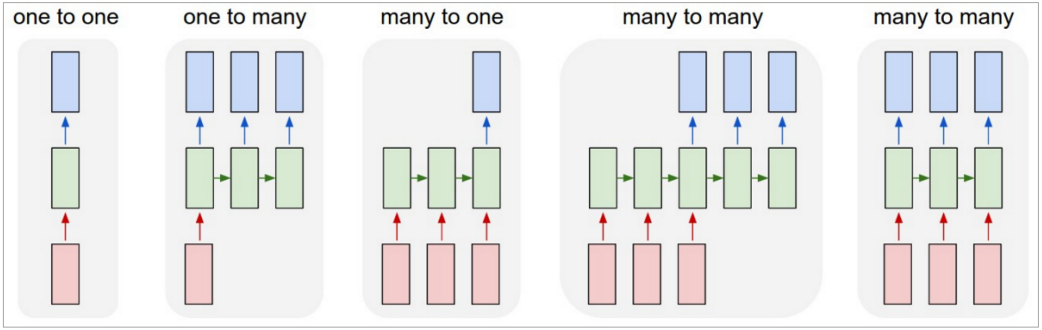


Foreshadowing: the Architecture of an RNN

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:

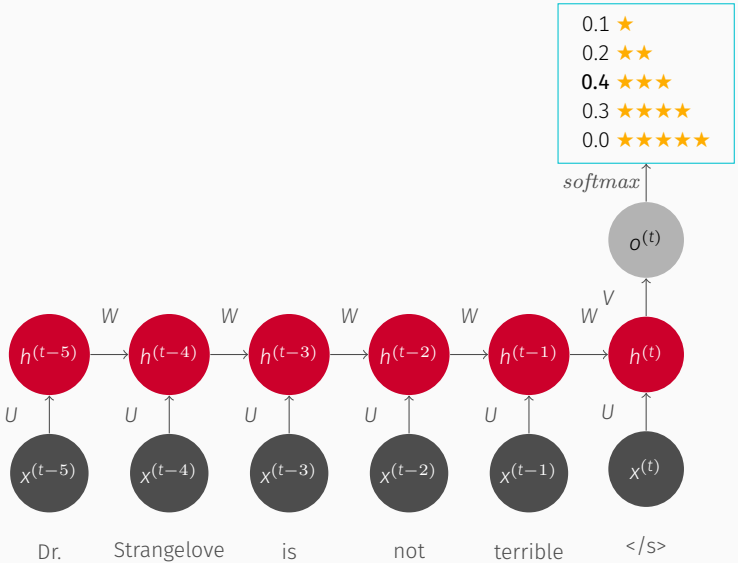


RNNs Can Be Used Various Ways, One of which is Classification



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.

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

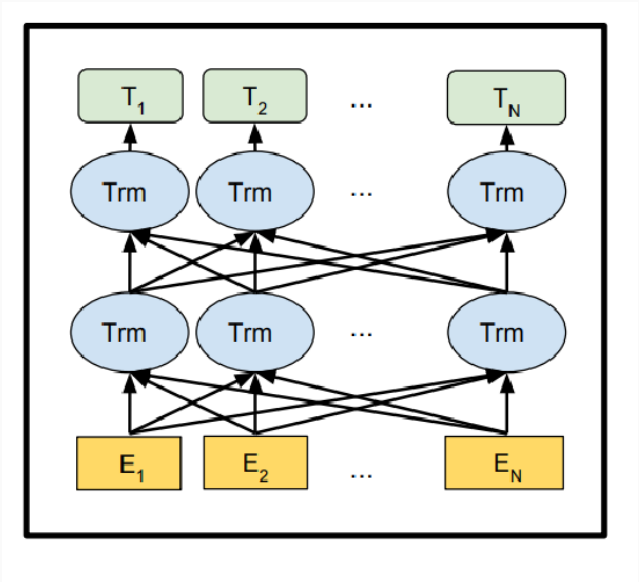
What is BERT?

- 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.

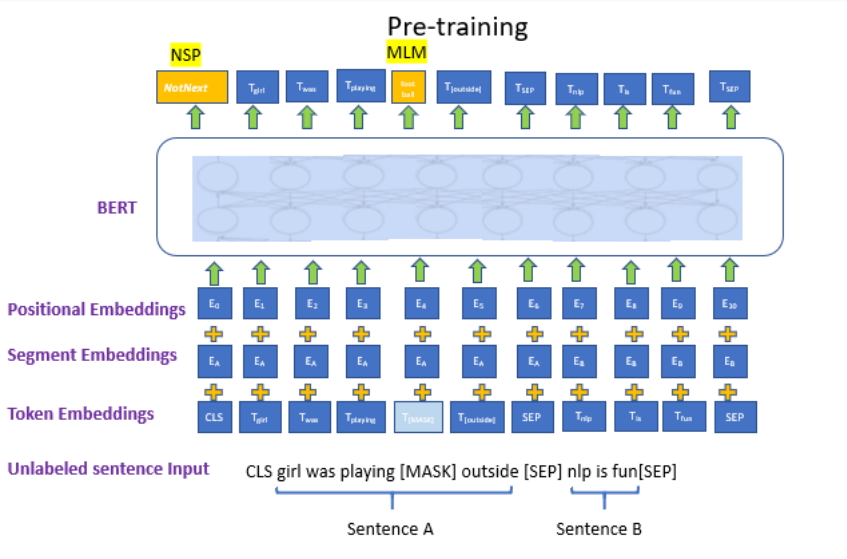
Overview: Advantages of BERT

- 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

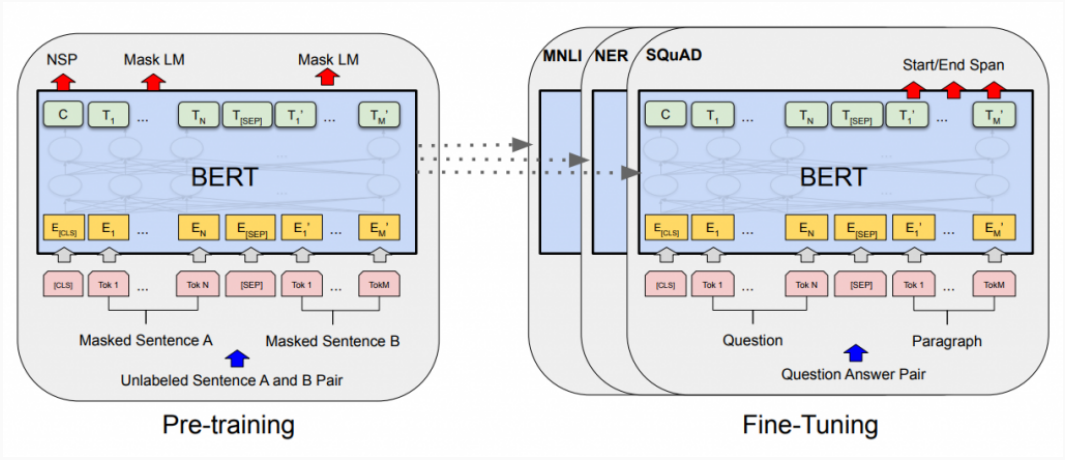


Some Tokens of Our Friendship

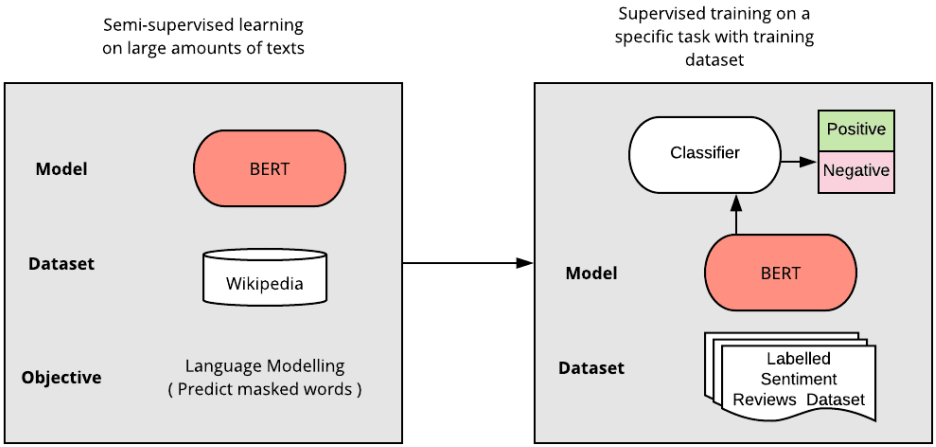


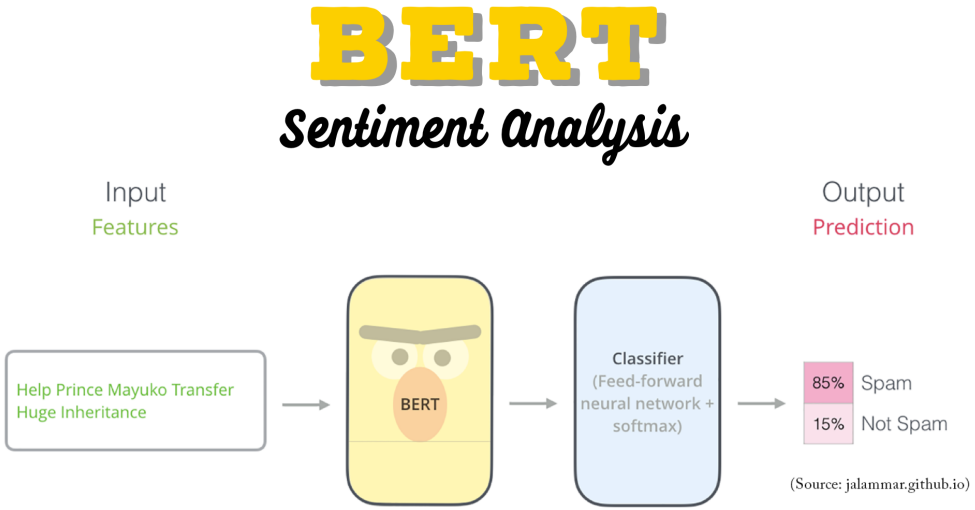
Fine Tuning BERT

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




BERT
Sentiment Analysis
Input
Features
Output
Prediction
Help Prince Mayuko Transfer
Huge Inheritance
BERT
Classifier
(Feed-forward
neural network +
softmax)
85%
Spam
15%
Not Spam
(Source: jalammar.github.io)

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?