NATURAL LANGUAGE PROCESSING

NATURAL LANGUAGE PROCESSING

- ✓NLP refers to a set of techniques involving the application of statistical methods, with or without insights from linguistics, to understand text for the sake of solving real-world tasks.

- ✓This "understanding" of text is mainly derived by:
  - ✓transforming texts to useable computational representations
    - ✓discrete or continuous combinatorial structures such as vectors or tensors, graphs, and trees.

TYPICAL APPLICATIONS IN NATURAL LANGUAGE PROCESSING [1]

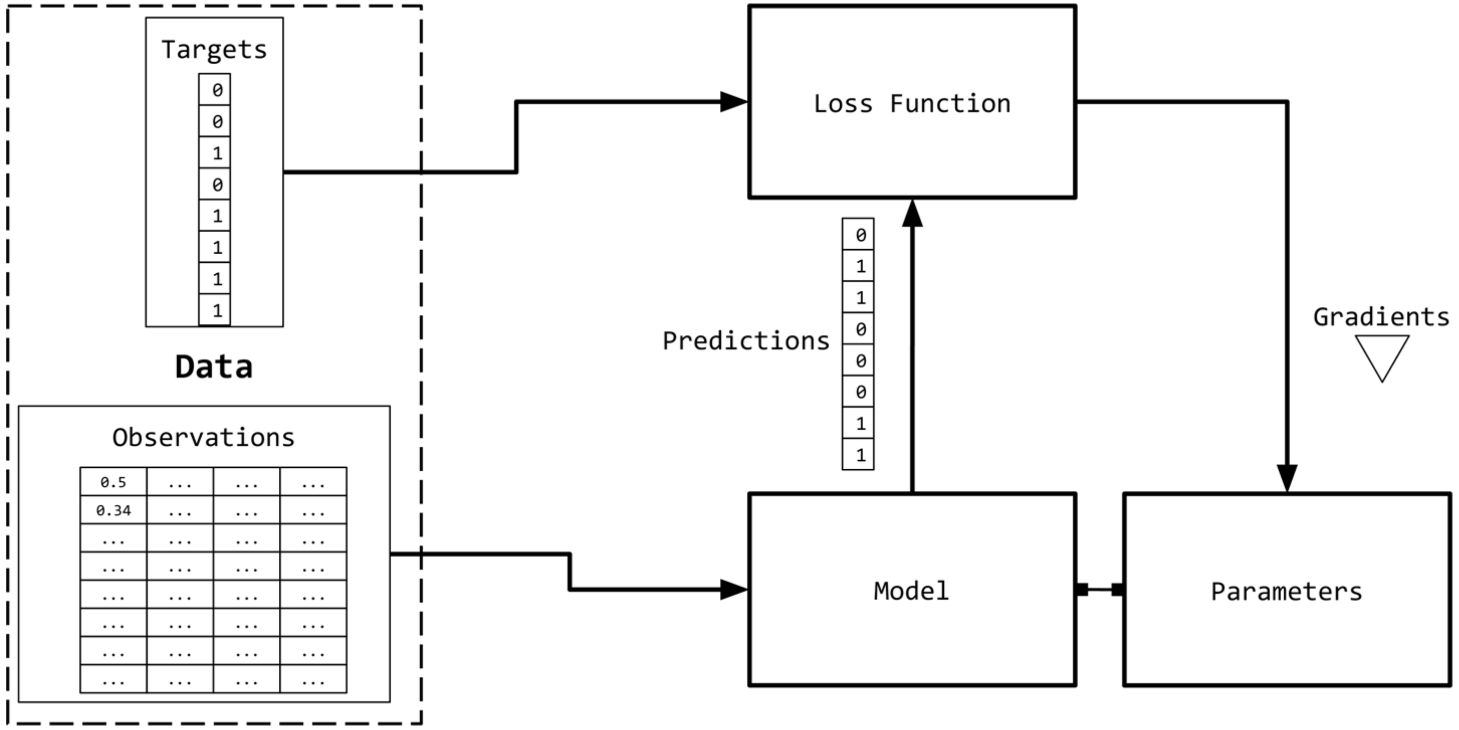
- ✓Text translations – one language to another
- ✓Text/Document Summarization
- ✓Topic Modeling
- ✓Question Answering
- ✓Automatic Transcription
  - ✓Speech to Text and vice-versa

TYPICAL APPLICATIONS IN NATURAL LANGUAGE PROCESSING [2]

- ✓Classification
  - ✓Assigning a grade or reading difficulty level to a piece of text
  - ✓Sentiment analysis
  - ✓Reviews and Ratings – restaurants, shops, movies, etc.
  - ✓Predict a user’s age from the tweet data
- ✓Language Modeling Problem
  - ✓Predict the next word, given the words seen in the past

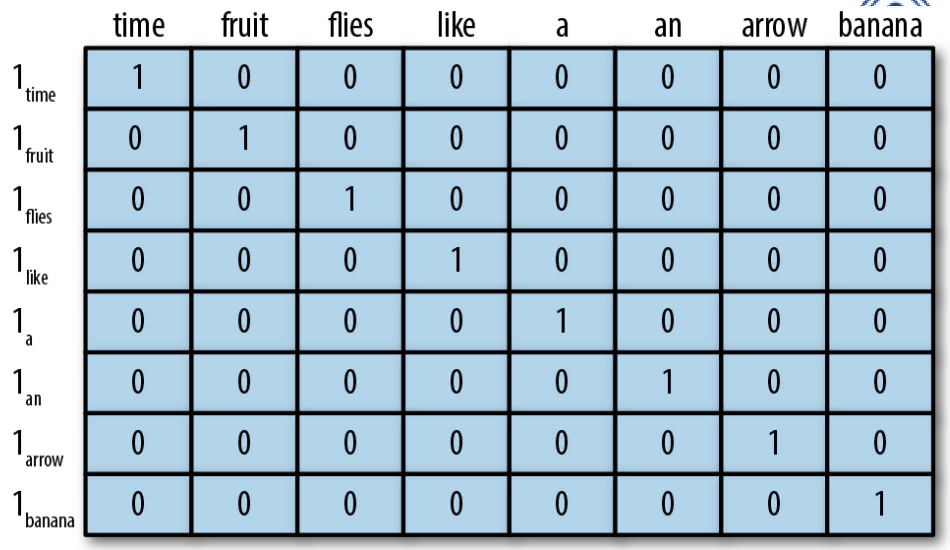
OBSERVATION AND TARGET ENCODING

- ✓Represent text to use them in conjunction with Machine Learning algorithms
- ✓Numerical vector
- ✓Learning a representation
- ✓Count-based representations
  - ✓Heuristic based
  - ✓Simple, but powerful



ONE-HOT REPRESENTATION

- ✓The one-hot representation starts with a zero vector, and sets as 1 the corresponding entry in the vector if the word is present in the sentence or document.
- ✓Consider the following two sentences:
  - ✓Time flies like an arrow.
  - ✓Fruit flies like a banana.
- ✓Tokenizing the sentences, ignoring punctuation, and treating everything as lowercase, will yield a vocabulary of size 8: {time, fruit, flies, like, a, an, arrow, banana}
- ✓The collapsed one-hot representation for a phrase, sentence, or a document is simply a **logical OR of the one-hot representations of its constituent words.**

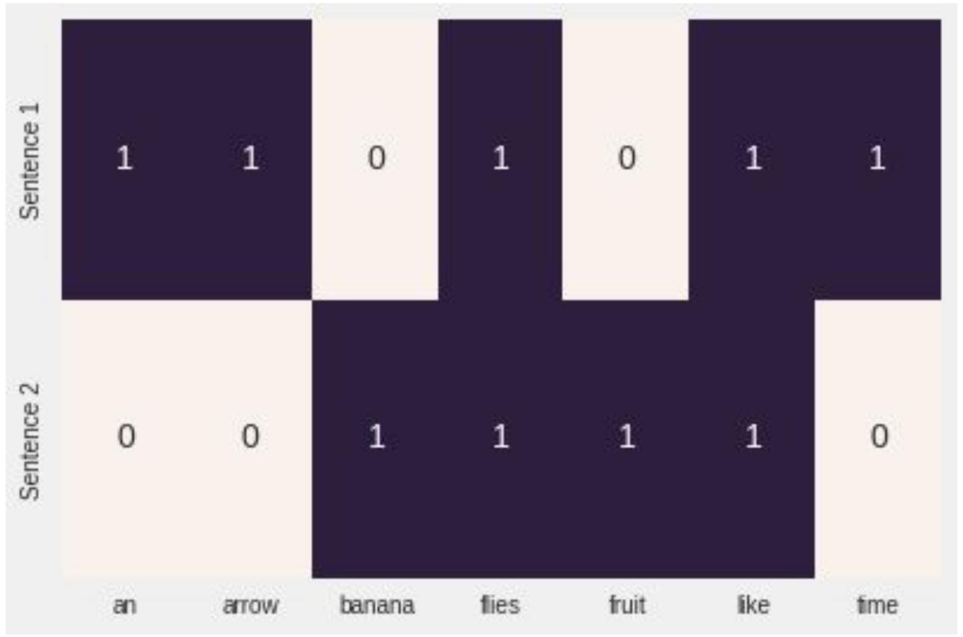


	time	fruit	flies	like	a	an	arrow	banana
1_{time}	1	0	0	0	0	0	0	0
1_{fruit}	0	1	0	0	0	0	0	0
1_{flies}	0	0	1	0	0	0	0	0
1_{like}	0	0	0	1	0	0	0	0
1_{a}	0	0	0	0	1	0	0	0
1_{an}	0	0	0	0	0	1	0	0
1_{arrow}	0	0	0	0	0	0	1	0
1_{banana}	0	0	0	0	0	0	0	1

TF (TERM FREQUENCY) REPRESENTATION

- ✓The **TF representation** of a phrase, sentence, or document is simply **the sum of the one-hot representations of its constituent words.**
- ✓The TF of the sentence "Fruit flies like time flies a fruit" has the following TF representation:
  - ✓[2, 2, 1, 1, 1, 0, 0, 0].
  - ✓{fruit, flies, like, time, a, an, arrow, banana}
- ✓Each **entry** is a **count** of the **number of times the corresponding word appears in the sentence (corpus).**
- ✓Denote TF of a word w by TF(w).

{an, arrow, banana, flies, fruit, like, time }



- Time flies like an arrow.
- Fruit flies like a banana.

TF-IDF REPRESENTATION [1]

- ✓Patent documents: {claim, system, method, procedure, …}
  - ✓ often repeated multiple times.
- ✓The TF representation weights words proportionally to their frequency.
- ✓To gain a deeper understanding of a patent, one needs to identify terms specific to the patent.
  - ✓Common words like "claim" do not add much to the understanding of a particular patent
  - ✓Rare words (for example, "tetrafluoroethylene") are more indicative of the nature of the patent document
- ✓Give a **larger weight to rare but relevant words** in the document representation

TF-IDF REPRESENTATION [2]

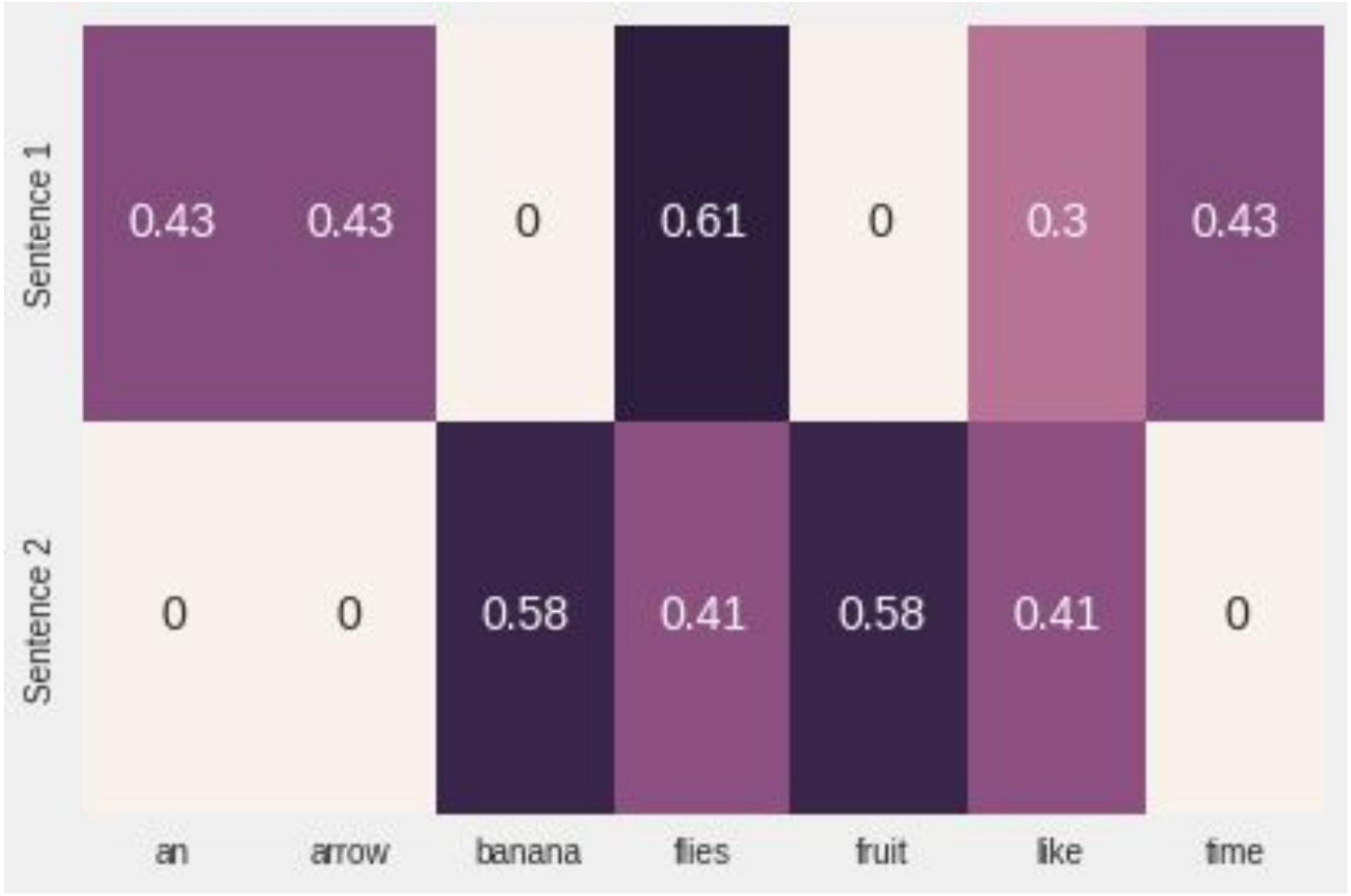
- ✓The Inverse Document Frequency (IDF) is a **heuristic** which is built to give **more weight to less common words**
- ✓The IDF representation **penalizes common tokens** and **rewards rare tokens** in the vector representation.
- ✓The IDF(w) of a token w is defined with respect to a corpus as:

$$IDF(w) = \log \frac{N}{n_w}$$

  - ✓where n_w is the no. of documents containing the word w and N is the total no. of documents.

- ✓The TF-IDF score is simply the product TF(w) * IDF(w)
  - ✓If a word is present in all documents then $N = n_w$ and IDF(w) = 0
  - ✓If a word is present only in one document, then IDF(w) is the maximum possible value i.e. logN

TF-IDF REPRESENTATION USING SCIKIT-LEARN

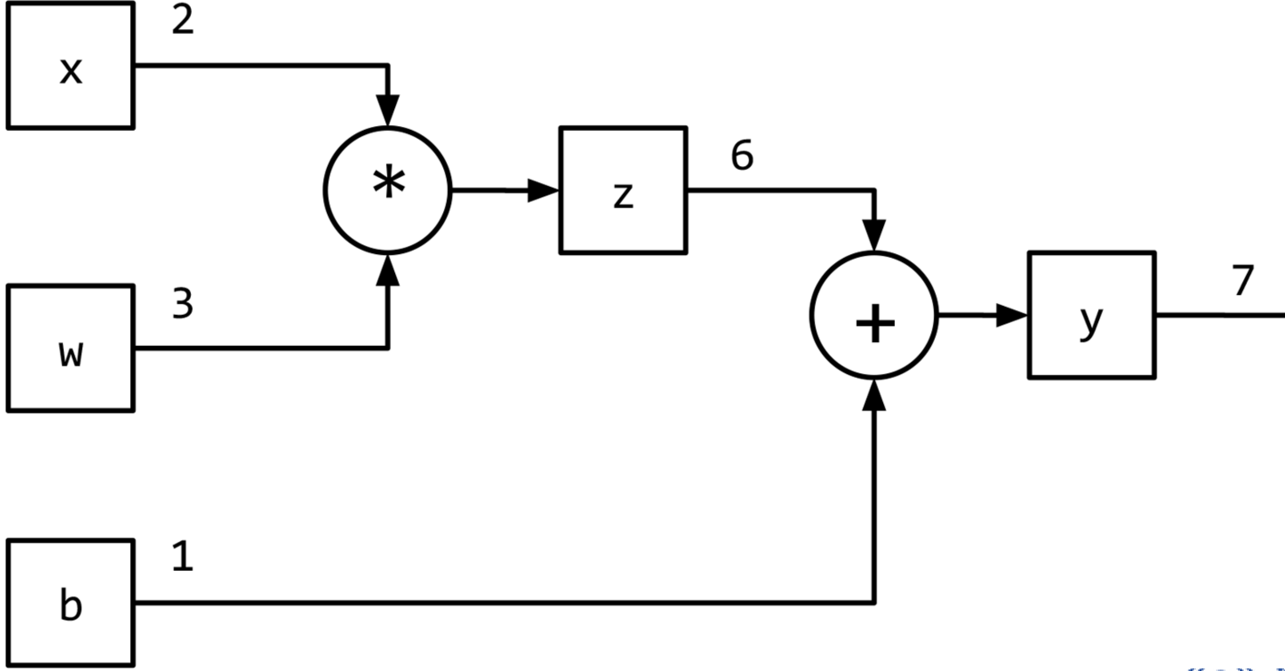


	an	arrow	banana	flies	fruit	like	time
Sentence 1	0.43	0.43	0	0.61	0	0.3	0.43
Sentence 2	0	0	0.58	0.41	0.58	0.41	0

TARGET ENCODING

- ✓The exact nature of the target variable depends on the NLP task being solved
- ✓One-hot encoding: machine translation, summarization, question answering
- ✓Categorical labels
  - ✓Unique index per label
  - ✓Problematic when the number of **output labels is simply too large**
- ✓Language Modeling problem
  - ✓Predict the **next word**, given the words seen in the past
  - ✓Label space is the entire vocabulary of the language
    - ✓Several hundred thousand
- ✓Predict a numerical value from a given text
  - ✓English essay – numerical grade
    - ✓Quality of the essay
    - ✓Readability score
  - ✓Rating from text reviews – movies, restaurants, places of tourism, etc.

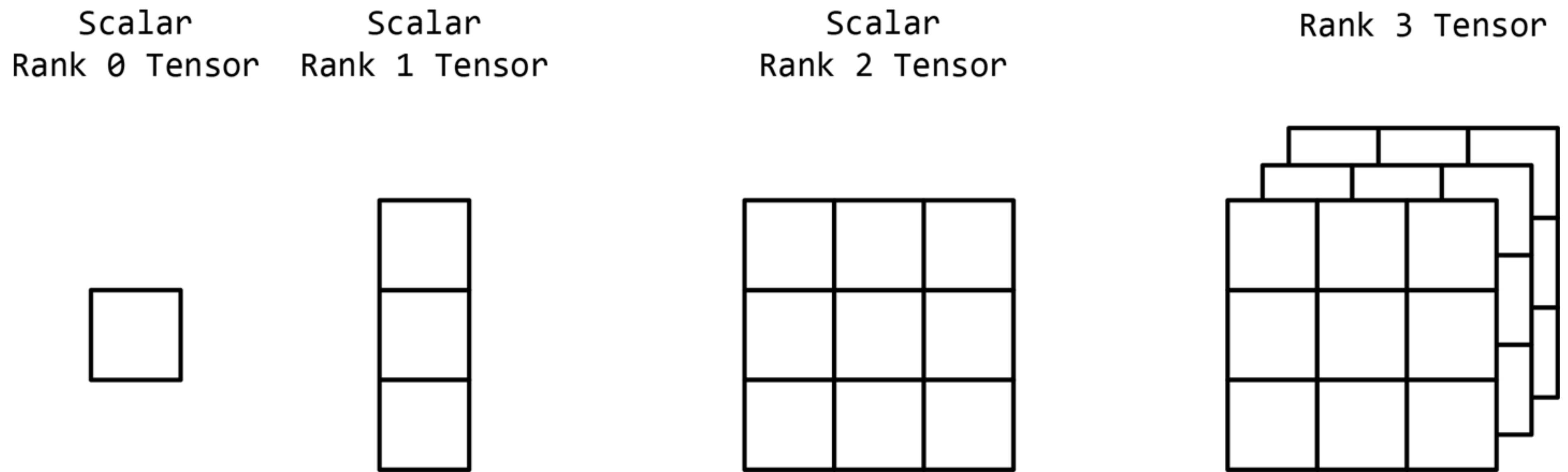
COMPUTATIONAL GRAPH



- ✓Data flow can be conveniently implemented using the **computational graph data structure.**
- ✓A computational graph is an abstraction that models mathematical expressions.
- ✓The implementations of the computational graph in Theano, TensorFlow, and PyTorch do additional bookkeeping:
  - ✓to implement **automatic differentiation** needed to obtain gradients of parameters during training in the supervised learning paradigm.
- ✓Modeling an expression in a computational graph: $y = wx + b$
- ✓*DAG* with nodes representing mathematical operations like multiplication and addition

PYTORCH BASICS

- ✓PyTorch will be used for implementing some deep learning models
- ✓PyTorch is an optimized **tensor manipulation library** that offers an array of packages for deep learning.
- ✓At the core of the library is the **tensor**, which is a mathematical object holding some multidimensional data.
  - ✓A tensor of order zero is just a number, or a scalar.
  - ✓A tensor of order one (1st order tensor) is an array of numbers, or a vector.
  - ✓A 2nd order tensor is an array of vectors, or a matrix.
- ✓A tensor can be generalized as an n-dimensional array of scalars



PYTORCH TENSOR OPERATIONS

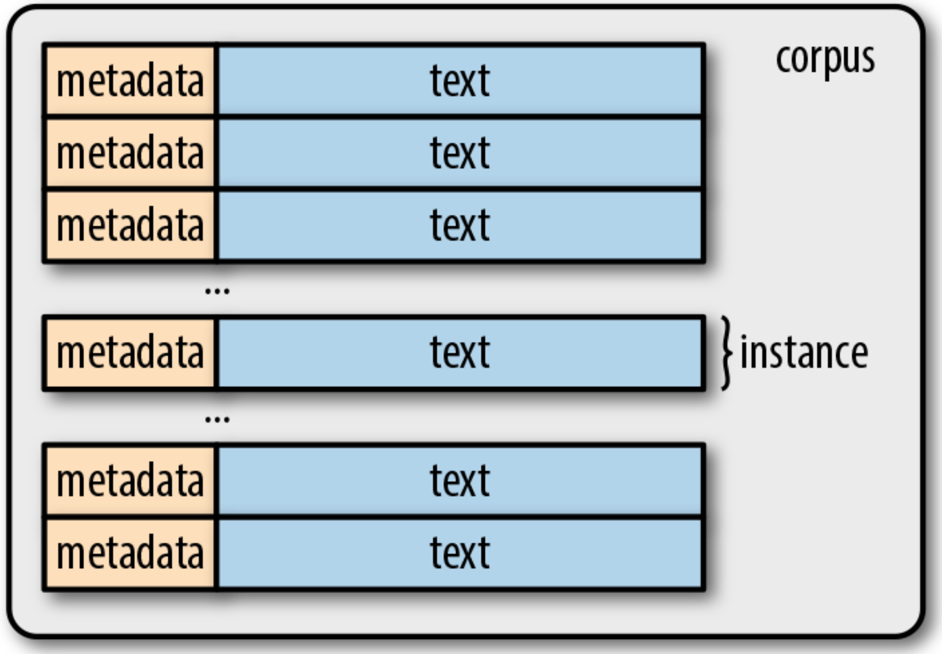
- Creating tensors
- Operations with tensors
- Indexing, slicing, and joining with tensors
- Computing gradients with tensors

NLP AND COMPUTATIONAL LINGUISTICS (CL)

- ✓Computational study of human language
  - ✓Natural language processing (NLP)
  - ✓Computational linguistics (CL)
- ✓NLP aims to develop methods for solving practical problems involving language, such as information extraction, automatic speech recognition, machine translation, sentiment analysis, question answering, and summarization.
- ✓CL is the study of the properties of human language by employing computational methods to understand its properties.
  - ✓How do we understand language?
  - ✓How do we produce language?
  - ✓How do we learn languages?
  - ✓What relationships do languages have with one another?
- ✓In practice, there is a lot of overlap between CL and NLP
- ✓Some subfields: phonology, morphology, syntax, semantics, pragmatics, …

CORPORA, TOKENS AND TYPES

- ✓A Corpus (plural: corpora) is the raw input dataset for all NLP methods
- ✓A corpus usually contains raw text (in ASCII or UTF8) and any metadata associated with the text.
- ✓The raw text is a sequence of characters (bytes), but most times it is useful to group those characters into contiguous units called tokens.
  - ✓In English, tokens correspond to words and numeric sequences separated by whitespace characters or punctuation.
- ✓The metadata could be any auxiliary piece of information associated with the text, like identifiers, labels, and timestamps.
- ✓The text along with its metadata is called an instance or data point.
- ✓The corpus, a collection of instances, is also known as a dataset.



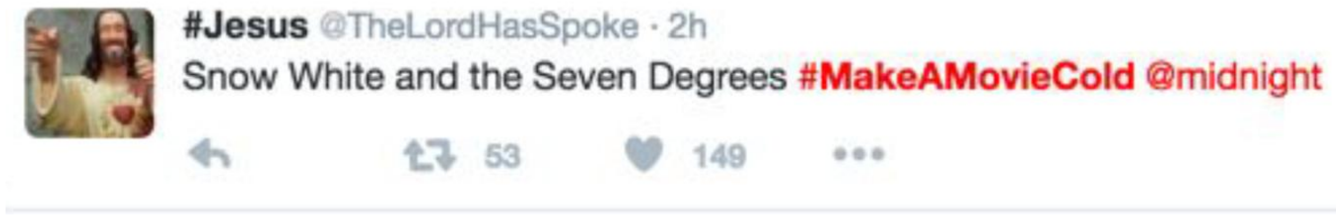
TOKENIZATION

- ✓The process of breaking a text down into tokens is called tokenization.
- ✓For example, there are six tokens in the Esperanto sentence “Maria frapis la verda sorĉistino.”
- ✓Tokenization can become more complicated than simply splitting text based on non-alphanumeric characters
- ✓For agglutinative languages like Turkish, splitting on whitespace and punctuation might not be sufficient, and more specialized techniques might be warranted.
- ✓Some neural networks entirely circumvent the issue of tokenization by representing text as a stream of bytes
  - ✓this becomes very important for agglutinative languages.

TOKENIZATION

Turkish	English
kork(-mak)	(to) fear
korku	fear
korkusuz	fearless
korkusuzlaş (-mak)	(to) become fearless
korkusuzlaşmış	One who has become fearless
korkusuzlaştır(-mak)	(to) make one fearless
korkusuzlaşstırıl(-mak)	(to) be made fearless
korkusuzlaştırılmış	One who has been made fearless
korkusuzlaştırılabil(-mek)	(to) be able to be made fearless
korkusuzlaştırılabilecek	One who will be able to be made fearless
korkusuzlaştırabileceklerimiz	Ones who we can make fearless
korkusuzlaştırabileceklerimizden	From the ones who we can make fearless
korkusuzlaştırabileceklerimizdenmiş	I gather that one is one of those we can make fearless
korkusuzlaştırabileceklerimizdenmişçesine	As if that one is one of those we can make fearless
korkusuzlaştırabileceklerimizdenmişçesineyken	when it seems like that one is one of those we can make fearless

TOKENIZATION



#Jesus @TheLordHasSpoke · 2h
Snow White and the Seven Degrees #MakeAMovieCold @midnight
53 149

- Tokenizing tweets involves preserving hashtags and @handles, and segmenting smiles such as :) and URLs as one unit.
- Should the hashtag #MakeAMovieCold be one token or four?
- Most research papers don't give much attention to these matters
- Many of the tokenization decisions tend to be arbitrary
- But those decisions can significantly affect accuracy in practice more than is acknowledged.
- Often considered the grunt work of preprocessing, most open source NLP packages provide reasonable support for tokenization.
- NLTK and spaCY are two commonly used packages for text processing

TYPES, ...

- Types are unique tokens present in a corpus.
- The set of all types in a corpus is its vocabulary or lexicon.
- Words can be distinguished as content words and stop-words.
- Stop-words are words as articles and prepositions serve mostly a grammatical purpose, like filler holding the content words.

FEATURE ENGINEERING

- The process of understanding the linguistics of a language and applying it to solving NLP problems is called feature engineering.
- Feature engineering is indispensable when building and deploying real world production systems.

UNIGRAMS, BIGRAMS, TRIGRAMS, …, N-GRAMS

- ✓N-grams are fixed-length (n) consecutive token sequences occurring in the text.
- ✓A bigram has two tokens, a unigram one, and so on.
- ✓Generating n-grams from a text is straightforward
  - ✓packages like spaCy and NLTK provide convenient methods.
- ✓Character n-grams
  - ✓Situations where the sub-word information itself carries useful information
  - ✓For example, the suffix “ol” in “methanol” indicates it is a kind of alcohol
  - ✓The information captured by the sub-word can be useful if the task is to classify organic compound names
  - ✓For such applications, we can reuse the same code, but treat every character n-gram as a token

LEMMAS AND STEMS [1]

- ✓Lemmas are root form of words.
- ✓Verb fly: flow, flew, flies, flown, flowing, and so on—
  - ✓fly is the lemma for all of these seemingly different words.
- ✓Lemmas are useful to reduce the tokens in order to keep the dimensionality of the vector representation small
- ✓This reduction process is called lemmatization
- ✓"Spacy" uses a predefined dictionary, called WordNet, for extracting lemmas.
- ✓Lemmatization can be framed as a machine learning problem requiring an understanding of the morphology of the language.

LEMMAS AND STEMS [2]

- ✓Stemming can be looked upon as the poor man's lemmatization.
- ✓It involves the use of handcrafted rules to strip endings of words to reduce them to a common form called stems.
- ✓Popular stemmers often implemented in open source packages include the Porter and Snowball stemmers.
- ✓"Geese"
  - ✓Lemmatization: goose
  - ✓Stemming: geesWe leave it to you to find the right
- ✓spaCy/NLTK haveAPIs to perform stemming.

CATEGORIZING WORDS: POS TAGGING

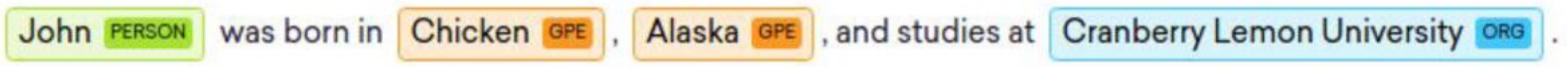
- ✓Categorizing or classifying documents is probably one of the earliest applications of NLP.
- ✓The TF and TF-IDF representations described earlier are immediately useful for classifying and categorizing longer chunks of text such as documents or sentences.
- ✓Problems that can be framed as supervised document classification problem include:
  - ✓assigning topic labels
  - ✓predicting sentiment of reviews
  - ✓filtering spam emails
  - ✓language identification
  - ✓email triaging
- ✓Semi-supervised versions, in which only a small labeled dataset is used, are also very useful

CATEGORIZING SPANS: CHUNKING AND NAMED ENTITY RECOGNITION [1]

- ✓Application often require labeling a span of text; that is, a contiguous multi-token boundary.
- ✓For example, consider the sentence, “Mary slapped the green witch.”
- ✓The noun phrases (NP) and verb phrases (VP) are identified as below:
  - ✓[NP Mary] [VP slapped] [the green witch].
- ✓This is called **chunking** or **shallow parsing.**
- ✓Shallow parsing aims to derive higher order units composed of the grammatical atoms, like nouns, verbs, adjectives, and so on.
- ✓It is possible to **write regular expressions** over the **parts-of-speech tags** to approximate shallow parsing if we lack data to train models for shallow parsing.
- ✓However, for English and other extensively spoken languages, such data and pretrained models exist.

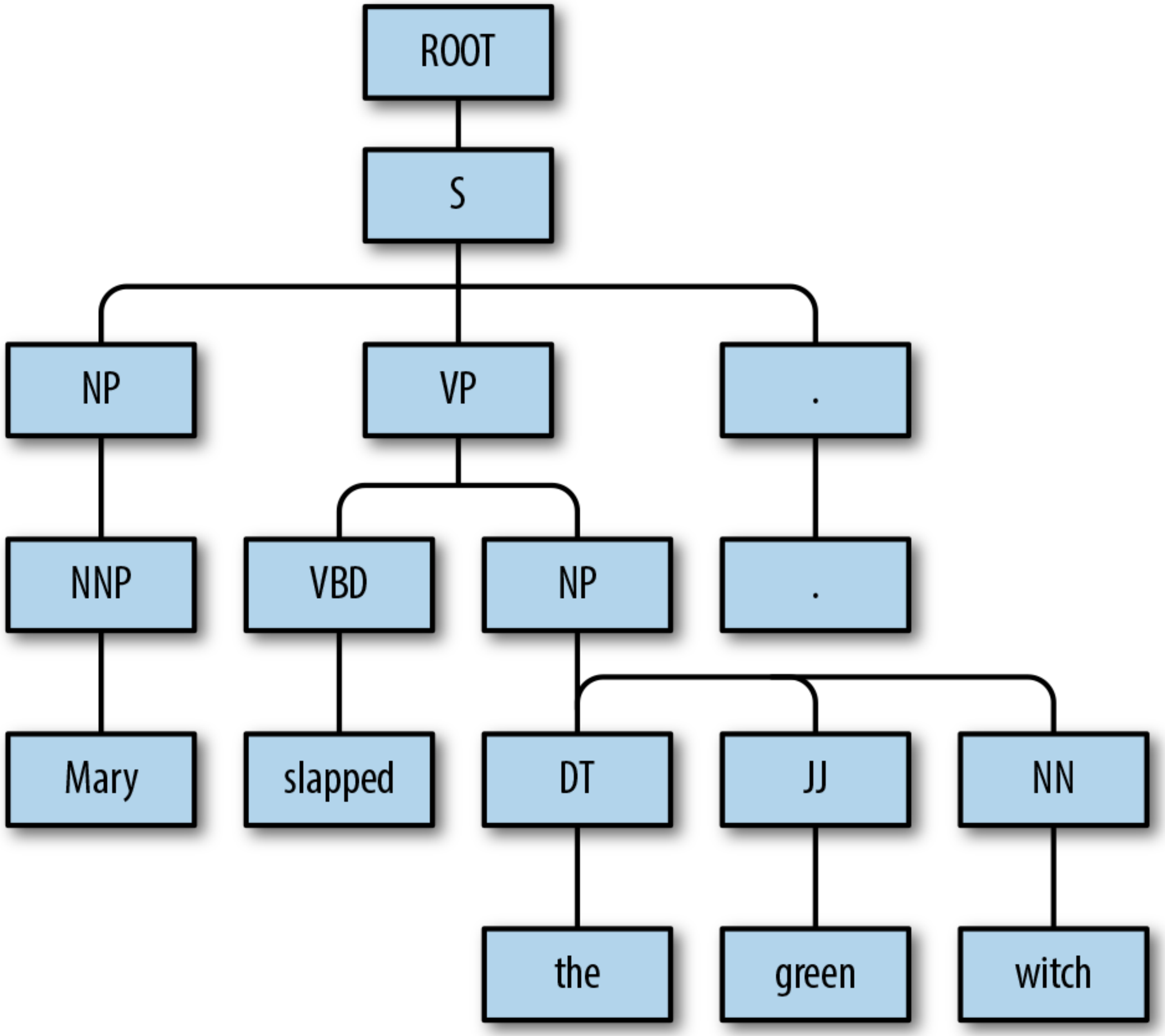
CATEGORIZING SPANS: CHUNKING AND NAMED ENTITY RECOGNITION [2]

- Another type of span that's useful is the named entity.
- A named entity is a string mention of a real world concept like a person, location, organization, drug name, and so on.
- Example:


John PERSON was born in Chicken GPE , Alaska GPE , and studies at Cranberry Lemon University ORG .

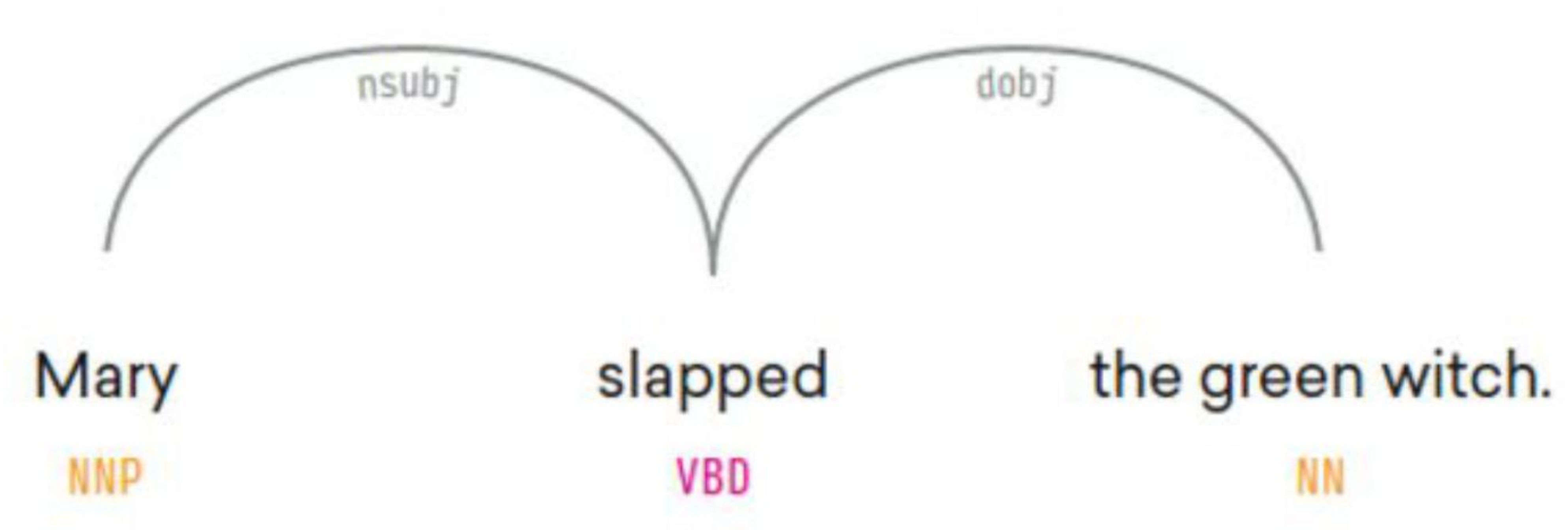
STRUCTURE OF SENTENCES

- Shallow parsing identifies phrasal units
- The task of identifying the relationship between them is called parsing.
- Parse trees indicate how different grammatical units in a sentence are related hierarchically.
- The parse tree in the adjoining figure is called a constituent parse for the sentence “Mary slapped the green witch.”



STRUCTURE OF SENTENCES

Dependency parsing is another useful way to depict relationships



WORD SENSES AND SEMANTICS

Very often words have more than one meanings, that differ based on the context.

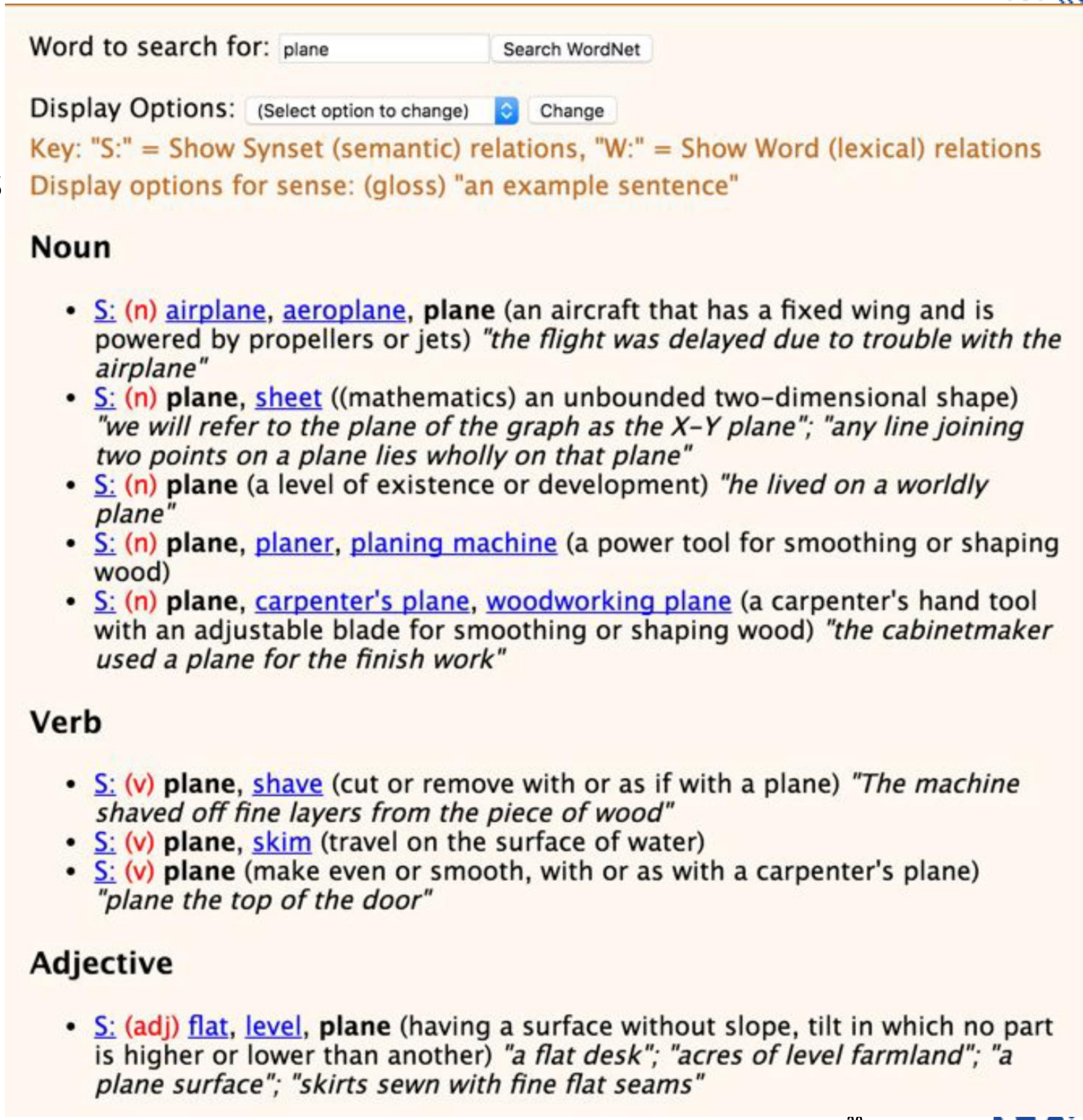
The different meanings of a word are called its “senses”.

WordNet, a long running lexical resource project from Princeton University, aims to catalog the senses of all (well, most) words in the English language, along with other lexical relationships.

The effort put in projects like WordNet can be usefully used for Natural Language Processing

Word senses can also be induced from the context.

Automatic discovery of word senses from text was actually the first place semi-supervised learning was applied to NLP.


Word to search for: plane Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) airplane, aeroplane, **plane** (an aircraft that has a fixed wing and is powered by propellers or jets) *"the flight was delayed due to trouble with the airplane"*
- S: (n) **plane**, sheet ((mathematics) an unbounded two-dimensional shape) *"we will refer to the plane of the graph as the X-Y plane"; "any line joining two points on a plane lies wholly on that plane"*
- S: (n) **plane** (a level of existence or development) *"he lived on a worldly plane"*
- S: (n) **plane**, planer, planing machine (a power tool for smoothing or shaping wood)
- S: (n) **plane**, carpenter's plane, woodworking plane (a carpenter's hand tool with an adjustable blade for smoothing or shaping wood) *"the cabinetmaker used a plane for the finish work"*

Verb

- S: (v) **plane**, shave (cut or remove with or as if with a plane) *"The machine shaved off fine layers from the piece of wood"*
- S: (v) **plane**, skim (travel on the surface of water)
- S: (v) **plane** (make even or smooth, with or as with a carpenter's plane) *"plane the top of the door"*

Adjective

- S: (adj) flat, level, **plane** (having a surface without slope, tilt in which no part is higher or lower than another) *"a flat desk"; "acres of level farmland"; "a plane surface"; "skirts sewn with fine flat seams"*

PREPROCESSING NATURAL LANGUAGE DATA [1]

- ✓Preprocessing of natural language data is required to make sure that the downstream modeling may be more accurate.
- ✓Common natural language preprocessing options include:
- ✓**Tokenization:** Splitting of a document (e.g. a book) into discrete elements of a language (e.g. words) called tokens
- ✓**Case conversion:** Disregarding any use of capitalization since in many cases, a capitalized word at the beginning of a sentence (e.g., She) has the same meaning as when it's used later in a sentence (she).
  - ✓Typically words are converted to lowercase
- ✓**Removal of Stop Words: F**requently occurring words that tend to contain relatively little distinctive meaning, such as **the**, **at**, **which**, and of.
  - ✓There is no universal consensus on the precise list of stop words
  - ✓Depends on the end application
  - ✓Suppose you build a model to classify movie reviews as positive or negative.
    - ✓Negations like didn't, isn't, and wouldn't
    - ✓Critical to identify the sentiment of the movie
    - ✓Shouldn't be removed as stop words

PREPROCESSING NATURAL LANGUAGE DATA [2]

- ✓Common natural language preprocessing options include:
- ✓**Removing Punctuation:** Punctuation marks generally don't add much value to a natural language model and so are often removed.
- ✓**Stemming:** Stemming is the truncation of words down to their stem.
  - ✓For example, the words house and housing both have the stem **hous.**
  - ✓Stemming can be productive with smaller datasets because it pools words with similar meaning into a single token.
  - ✓Stemming helps techniques like word2vec or GloVe to more accurately identify an appropriate location for the token in word-vector space
- ✓**Handling n-grams:** Some words commonly co-occur in such a way that the com- bination of words is better suited to being considered a single concept than several separate concepts.
  - ✓For example, New York is a bigram (an n-gram of length two), and New York City is a trigram (an n-gram of length three).
  - ✓When chained together, the words new, york, and city have a specific meaning that might be better captured by a single token (and therefore a single location in word-vector space) than three separate ones

PREPROCESSING NATURAL LANGUAGE DATA [3]

- ✓Should you always use all pre-processing steps for all NLP tasks?
- ✓Some of the data preprocessing steps might be task specific.
- ✓A data scientist should use her intuition to weigh whether a particular step might ultimately be valuable to her downstream task
- ✓Stemming may be helpful for a small corpus but unhelpful for a large one.
- ✓Converting all characters to lowercase is likely to be helpful in a small corpus, but, in a larger corpus that has many more examples of individual uses of words, the distinction of the use of capital letters might be valuable
  - ✓For example, say, general (an adjective meaning "widespread") versus General (a noun meaning the commander of an army)
- ✓Better not to remove punctuations while building a question-answering algorithm
  - ✓It could use question marks to help identification of questions.
- ✓Negations may be helpful as stop words for some classifiers but probably not for a sentiment classifier, for example.
  - ✓In many applications, it is recommended to remove only a limited number of stop words.

PREPROCESSING NATURAL LANGUAGE DATA [4]

- ✓To determine whether a given preprocessing step may be helpful or not, one can investigate the situation empirically by incorporating the step and observing whether it impacts the accuracy of your deep learning model downstream.
- ✓As a general rule, the larger a corpus becomes, the number of preprocessing steps that will be helpful reduces.
- ✓For a small corpus,
  - ✓There is a concern about encountering words that are rare or
  - ✓That are outside the vocabulary of your training dataset.
  - ✓By pooling several rare words into a single common token, its is more likely to train a model effectively on the meaning of the group of related words.
- ✓With a larger corpus, rare and out-of- vocabulary words become less and less of an issue.
  - ✓avoid pooling several words into a single common token is more helpful
  - ✓because there will be enough instances of even the less-frequently-occurring words to
    - ✓effectively model their unique meaning
    - ✓as well as to model the relatively subtle nuances between related words (that might otherwise have been pooled together).

NLTK

NLTK is a leading platform for building Python programs to work with human language data.

It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet.

Suite of text processing libraries for:

- Classification
- Tokenization
- Stemming
- Tagging
- Parsing, and semantic reasoning,
- Wrappers for industrial-strength NLP libraries

GENSIM

- ✓**Gensim** is an open-source library for unsupervised topic modeling and natural language processing, using modern statistical machine learning.
- ✓Gensim is implemented in Python and Cython for performance.
- ✓Gensim is designed to handle large text collections using data streaming and incremental online algorithms,
- ✓This differentiates it from most other machine learning software packages that target only in-memory processing.
- ✓Gensim includes streamed parallelized implementations of
  - ✓fastText word2vec and doc2vec algorithms,
  - ✓latent semantic analysis (LSA, LSI, SVD)
  - ✓non-negative matrix factorization (NMF)
  - ✓latent Dirichlet allocation (LDA)
  - ✓tf-idf and random projections.

BOKEH

- Bokeh is an interactive visualization library for modern web browsers.
- It provides elegant, concise construction of versatile graphics, and affords high-performance interactivity over large or streaming datasets.
- Bokeh can help in the rapid and easy development of interactive plots, dashboards, and data applications.

NLP PRE-PROCESSING

A PREPROCESSING EXAMPLE [1]

- A small corpus of out-of-copyright books from Project Gutenberg
- Available within NLTK for easy download
- The corpus consists of 18 literary works, including Jane Austen's Emma, Lewis Carroll's Alice in Wonderland, and three plays by William Shakespeare.
- The corpus comes to around 2.6 million words

A PREPROCESSING EXAMPLE [2]

- ✓To extract words and sentences from the corpus, we use Tokenization
- ✓Nltk's list of stop words is used to remove frequently occurring stop words in the sentence.
- ✓Stemming is done using Porter's algorithm provided by nltk
- ✓N-grams are handled by the Phrases() and Phraser() methods from the genism library
  - ✓It is possible to identify 4-grams and 5-grams but they do not add much value to the whole process
  - ✓Hence, in practice 2-grams and 3-grams are sufficient
- ✓Stop-word removal and stemming have not been implemented here.
- ✓Word embeddings are created using word2vec

CREATING WORD EMBEDDINGS WITH WORD2VEC [1]

- ✓ Intuitive understanding of word vectors
- ✓ A given word's meaning can well be represented as the average of the words that tend to occur around it.
- ✓ Word2vec is an unsupervised learning technique
  - ✓ Applied to a corpus of words without any labels
- ✓ Two underlying architectures for word2vec are skip-gram(SG) and continuous bag of words(CBOW)
- ✓ Produce roughly comparable results despite maximizing probabilities from the opposite perspectives
- ✓ **Sentence:** "We were running late for the conference call."
- ✓ **Target word:** late
- ✓ **Window size:** three words
- ✓ **Context words:** three words to the left and right of "late"

CREATING WORD EMBEDDINGS WITH WORD2VEC [2]

- ✓ Sentence: “We were running late for the conference call.”
- ✓ Target word: late
- ✓ Context words: {we, were, running}, {for the conference}
- ✓ SG (skip-gram) architecture
  - ✓ Predicts the context word given the target words
- ✓ CBOW (continuous bag of words) architecture
  - ✓ The target word is predicted based on the context words
- ✓ Approach from opposite perspectives

CREATING WORD EMBEDDINGS WITH WORD2VEC [3]

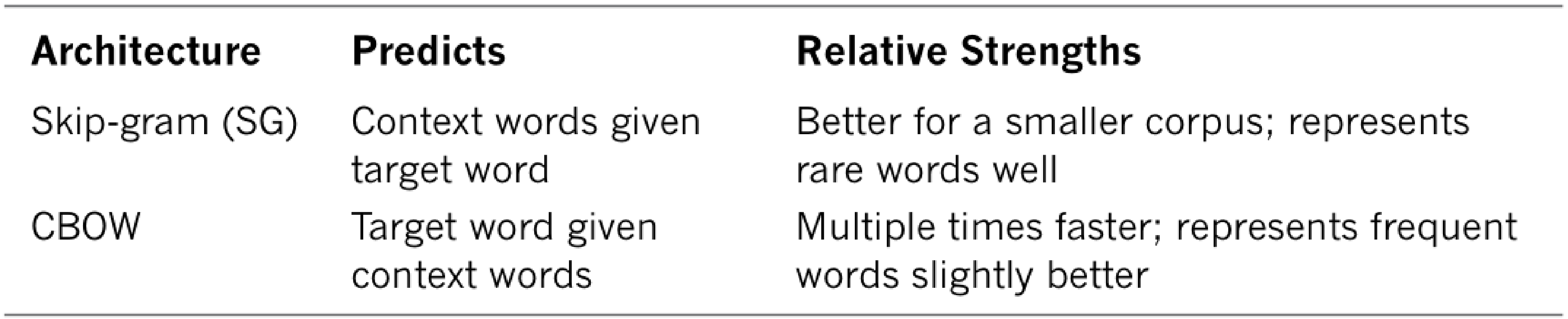
- **CBOW (Continuous Bag of Words) architecture**
- target word is predicted to be the average of all the context words considered jointly (simultaneously)
- Position of context words i.e. their order and also whether the context word occurs before or after the target word is not considered
- The term "bag of words" implies that the sequence is irrelevant
- The target word is estimated by calculating the average of all the context words contained in the bag
- The order of words would matter if the concern were syntax i.e. the grammar of language
- When concerned only with the semantics, i.e. meaning of words, it turns out that the order of context words is, on an average, irrelevant.

CREATING WORD EMBEDDINGS WITH WORD2VEC [4]

- **CBOW architecture**
- The term "continuous" in CBOW refers to the fact that "word window" slide continuously one word at a time from the first word of the corpus all the way through to the final word.
- At each position → the target word is estimated given the context words.
- Stochastic gradient descent is used to shift the location of words within vector space.
- Leads to a gradual improvement in the target-word estimates

CREATING WORD EMBEDDINGS WITH WORD2VEC [5]

- ✓For small corpus, SG architecture is a better choice
  - ✓Represents rare words in word-vector space well
  - ✓Better choice with a small corpus
- ✓CBOW is computationally more efficient
  - ✓Represents frequently occurring words slightly better
  - ✓Better for large corpus



Architecture	Predicts	Relative Strengths
Skip-gram (SG)	Context words given target word	Better for a smaller corpus; represents rare words well
CBOW	Target word given context words	Multiple times faster; represents frequent words slightly better

CREATING WORD EMBEDDINGS WITH WORD2VEC [6]

- ✓A major alternative to word2vec is GloVe—global vectors for word representation
  - ✓Introduced by the prominent natural language researchers Jeffrey Pennington, Richard Socher, and Christopher Manning
  - ✓GloVe is count based
  - ✓designed to be parallelized over multiple processors or even multiple machines
  - ✓Good option when dealing with a word-vector space with many unique words and a very large corpus
- ✓Another leading alternative to both word2vec and GloVe is fastText.
  - ✓developed by researchers at Facebook.
  - ✓A major benefit of fastText is that it operates on a subword level
    - ✓its "word" vectors are actually subcomponents of words.
  - ✓This enables fastText to work around some of the issues related to rare words and out-of-vocabulary words

EVALUATION OF WORD VECTORS [1]

- ✓Two broad perspectives for evaluating the quality of word vectors: intrinsic and extrinsic evaluations.
- ✓Extrinsic evaluation
- ✓Assess the performance of your word vectors within the downstream NLP application of interest
- ✓For example, sentiment-analysis classifier, or named-entity recognition tool.
- ✓Might take a longer time to carry out
  - ✓All downstream processing steps have to be completed
  - ✓Training a computationally intensive deep learning model
- ✓Higher confidence with respect to the appropriateness of structure and changes of word vectors if they relate to an appreciable improvement in the accuracy of your NLP application.

EVALUATING WORD VECTORS [2]

$$V_{king} - V_{man} + V_{woman} = V_{queen}$$
$$V_{bezos} - V_{amazon} + V_{tesla} = V_{musk}$$
$$V_{windows} - V_{microsoft} + V_{google} = V_{android}$$

- ✓Intrinsic evaluations involve assessing the performance of the word vectors on some specific intermediate sub-task.
- ✓One common such task is assessing whether your word vectors correspond well to arithmetical analogies.
  - ✓For example, if you start at the word-vector location for king, subtract man, and add woman, do you end up near the word-vector location for queen?
- ✓Intrinsic evaluations are faster than extrinsic evaluations
- ✓May help better understand (and therefore troubleshoot) intermediate steps within the broader NLP process.
- ✓Intrinsic evaluations are limited in that:
  - ✓they may not ultimately lead to improvements in the accuracy of your NLP application downstream
- ✓Require identification of a reliable, quantifiable relationship between performance on the intermediate test and the final NLP application.

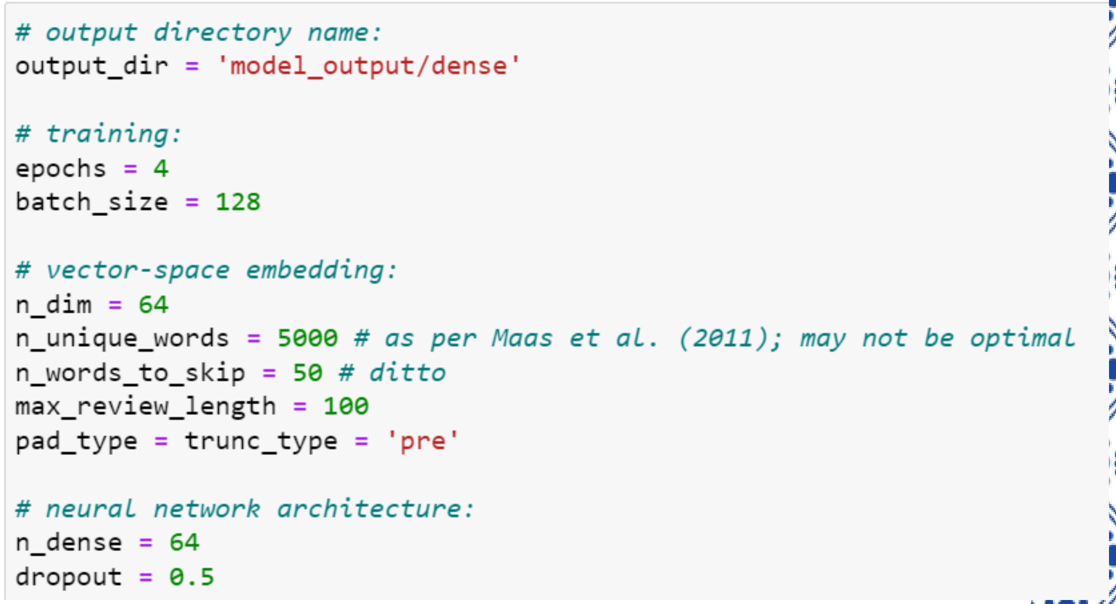
CLASSIFYING FILM REVIEWS

CLASSIFYING IMDB FILM REVIEWS USING NLP AND DEEP LEARNING [1]

- ✓A binary classifier that predicts whether a given film review is a positive one or a negative one
- ✓Train and test a relatively simple dense network
- ✓New dependencies include packages for:
  - ✓loading a dataset of film reviews
  - ✓saving model parameters as we train,
  - ✓Calculating ROC AUC
- ✓Good programming practice to put hyperparameters at the top of your file.
  - ✓This makes it easier to experiment with these hyperparameters.
  - ✓Easy to understand the code
  - ✓Modify it if required later

CLASSIFYING IMBD FILM REVIEWS USING NLP AND DEEP LEARNING [2]

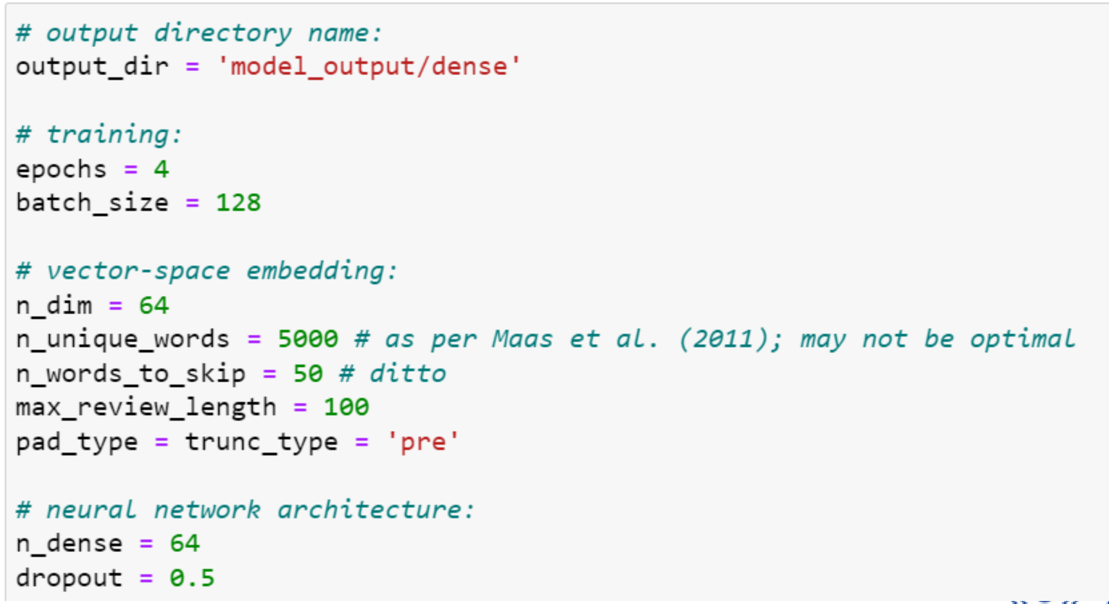
- ✓Hyperparameters
- ✓**output_dir:** A directory name (ideally, a unique one) to store the model's parameters after each epoch
  - ✓allowing us to return to the parameters from any epoch of our choice at a later time.
- ✓**epochs:** The number of training epochs
  - ✓NLP models often overfit to the training data in fewer epochs than machine vision models.
- ✓**batch_size:** The number of training examples used during each round of model training
- ✓**n_dim:** The number of dimensions in the word-vector space.



```
# output directory name:
output_dir = 'model_output/dense'

# training:
epochs = 4
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 5000 # as per Maas et al. (2011); may not be optimal
n_words_to_skip = 50 # ditto
max_review_length = 100
pad_type = trunc_type = 'pre'

# neural network architecture:
n_dense = 64
dropout = 0.5
```

CLASSIFYING IMBD FILM REVIEWS USING NLP AND DEEP LEARNING [3]

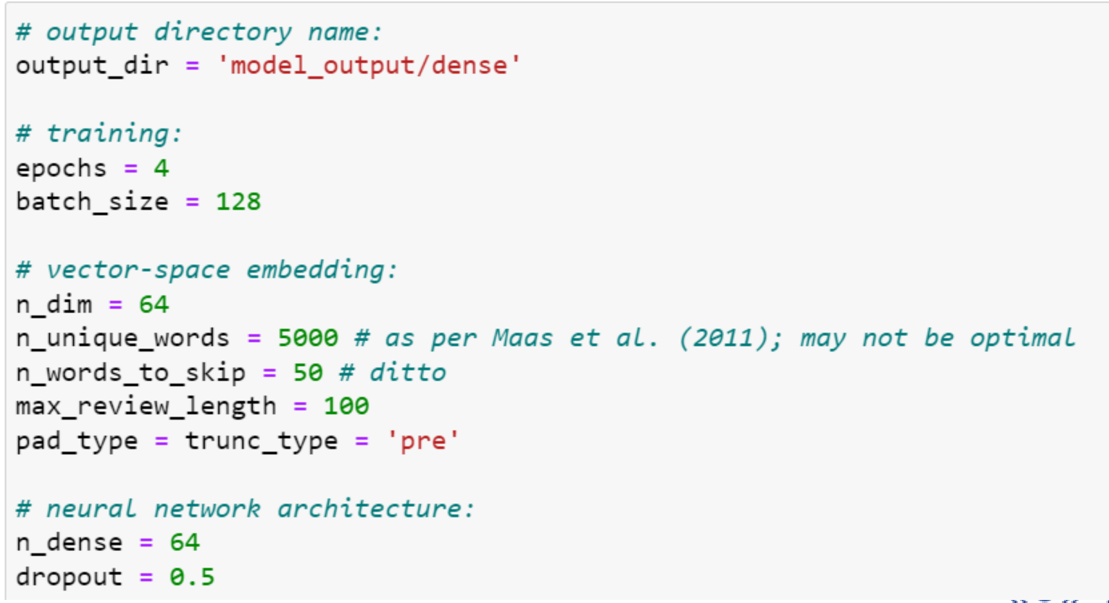
- ✓**n_unique_words:**
- ✓In our earlier word2vec exercise we included a token in the word-vector vocabulary only if they occurred at least a certain number of times within our corpus.
- ✓An alternative approach is to sort all of the tokens in the corpus by the number of times they occur.
- ✓The only use a certain number of the most popular words.
- ✓Andrew Maas and his coworkers opted to use the 5,000 most popular words across their film-review corpus
- ✓Here, we also use 5,000 most popular words



```
# output directory name:
output_dir = 'model_output/dense'

# training:
epochs = 4
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 5000 # as per Maas et al. (2011); may not be optimal
n_words_to_skip = 50 # ditto
max_review_length = 100
pad_type = trunc_type = 'pre'

# neural network architecture:
n_dense = 64
dropout = 0.5
```

CLASSIFYING IMBD FILM REVIEWS USING NLP AND DEEP LEARNING [4]

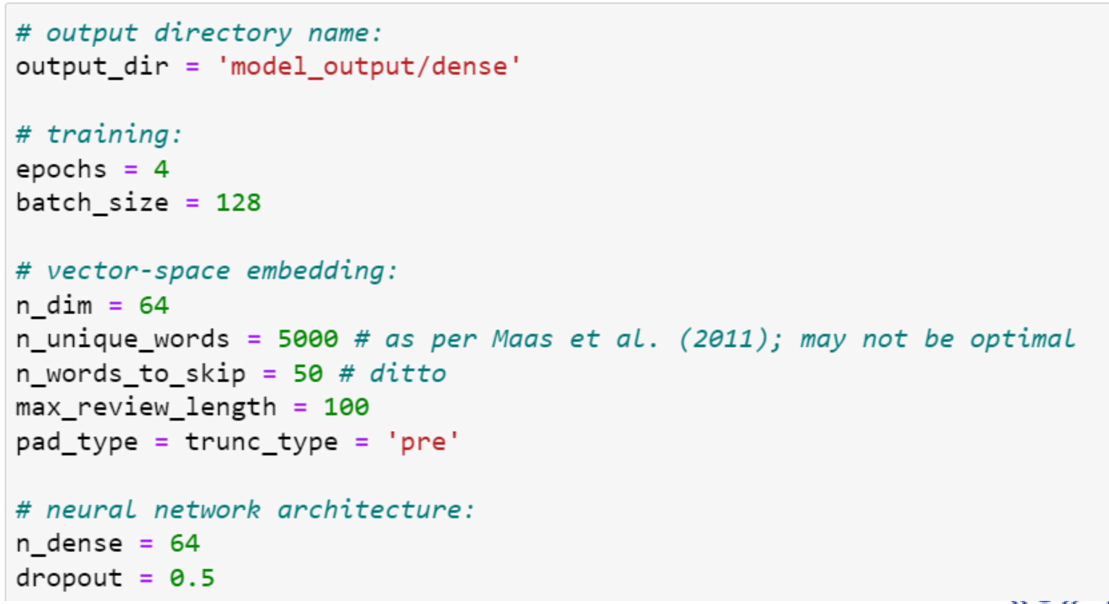
- ✓**n_words_to_skip:**
  - ✓Alternative approach to removing stop words
  - ✓Usual approach is to remove stop words from a manually curated list of stop word
  - ✓Assume that the 50 most frequently occurring words across the film-review corpus would serve as a decent list of stop words.
- ✓**max_review_length:**
  - ✓Each movie review must have the same length so that TensorFlow knows the shape of the input data that will be flowing through the deep learning model.
  - ✓Here, a review length of 100 words was used
  - ✓Any reviews longer than 100 are truncated.
  - ✓Any reviews shorter than 100 are padded with a special padding character



```
# output directory name:
output_dir = 'model_output/dense'

# training:
epochs = 4
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 5000 # as per Maas et al. (2011); may not be optimal
n_words_to_skip = 50 # ditto
max_review_length = 100
pad_type = trunc_type = 'pre'

# neural network architecture:
n_dense = 64
dropout = 0.5
```

CLASSIFYING IMBD FILM REVIEWS USING NLP AND DEEP LEARNING [5]

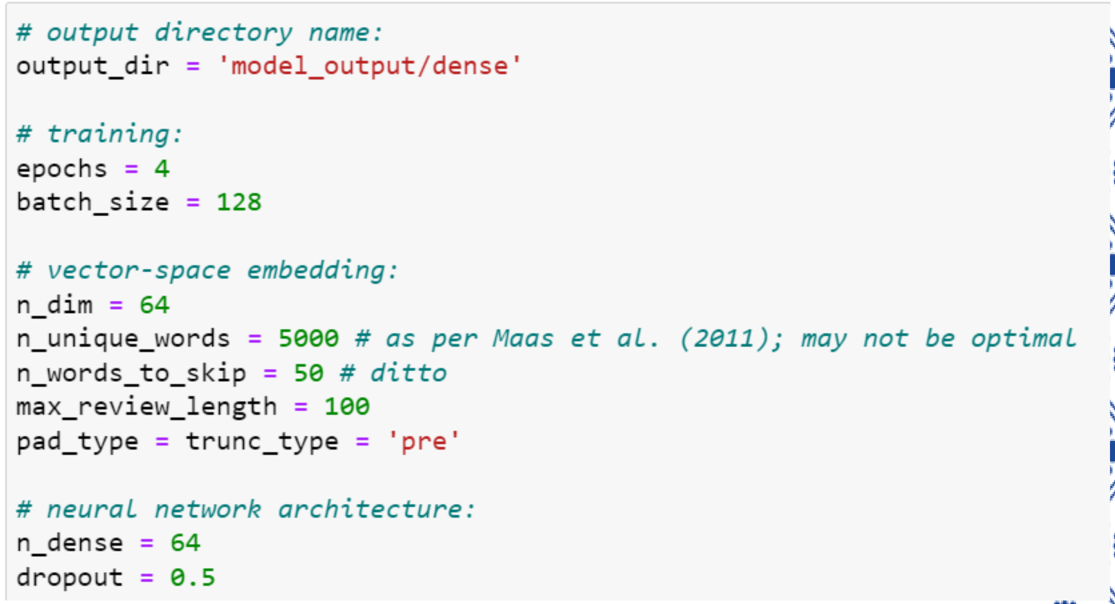
- ✓**pad_type:**
  - ✓Padding characters can be added either at the start or at the end of each review
  - ✓'pre’: add padding characters to the start of every review.
  - ✓'post’: adds padding characters to the end
  - ✓The location of padding characters doesn’t make much difference with a dense network
  - ✓For sequential-data layer types, it’s generally best to use 'pre' because the content at the end of the document is more influential in the model
  - ✓largely uninformative padding characters to be at the beginning of the document.



```
# output directory name:
output_dir = 'model_output/dense'

# training:
epochs = 4
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 5000 # as per Maas et al. (2011); may not be optimal
n_words_to_skip = 50 # ditto
max_review_length = 100
pad_type = trunc_type = 'pre'

# neural network architecture:
n_dense = 64
dropout = 0.5
```

CLASSIFYING IMBD FILM REVIEWS USING NLP AND DEEP LEARNING [5]

- ✓ **trunc_type:**
  - ✓ truncation options are 'pre' or 'post'.
  - ✓ 'pre' will remove words from the beginning of the review, whereas 'post' will remove them from the end.
  - ✓ If we select 'pre', we're making a bold assumption that the end of film reviews tend to include more information on review sentiment than the beginning.
- ✓ **n_dense:** The number of neurons to include in the dense layer of the neural network architecture.
  - ✓ Single layer of dense network
  - ✓ Experimentation for Optimization
- ✓ **dropout:** Selected as 0.5
  - ✓ Can be explored through experimentation



```
# output directory name:
output_dir = 'model_output/dense'

# training:
epochs = 4
batch_size = 128

# vector-space embedding:
n_dim = 64
n_unique_words = 5000 # as per Maas et al. (2011); may not be optimal
n_words_to_skip = 50 # ditto
max_review_length = 100
pad_type = trunc_type = 'pre'

# neural network architecture:
n_dense = 64
dropout = 0.5
```

LOADING THE IMDB FILM REVIEW DATA

- ✓The dataset made up of the natural language of reviews from the publicly available Internet Movie Database (IMDb; imdb.com).
- ✓Consists of 50,000 reviews
- ✓Equally divided among training (x_train) and validation (x_valid)
- ✓Users
  - ✓Submit a review of a given film in text
  - ✓Provide a star rating, with a maximum of 10 stars.
- ✓The labels (y_train and y_valid) are binary, based on these star ratings:
  - ✓Reviews with a score of four stars or fewer are considered to be a negative review (y = 0).
  - ✓Reviews with a score of seven stars or more, are classed as a positive review (y = 1).
  - ✓Moderate reviews—those with five or six stars—are not included in the dataset, making the binary classification task easier for any model

```
(x_train, y_train), (x_valid, y_valid) = \
    imdb.load_data(num_words=n_unique_words, skip_top=n_words_to_skip)
```

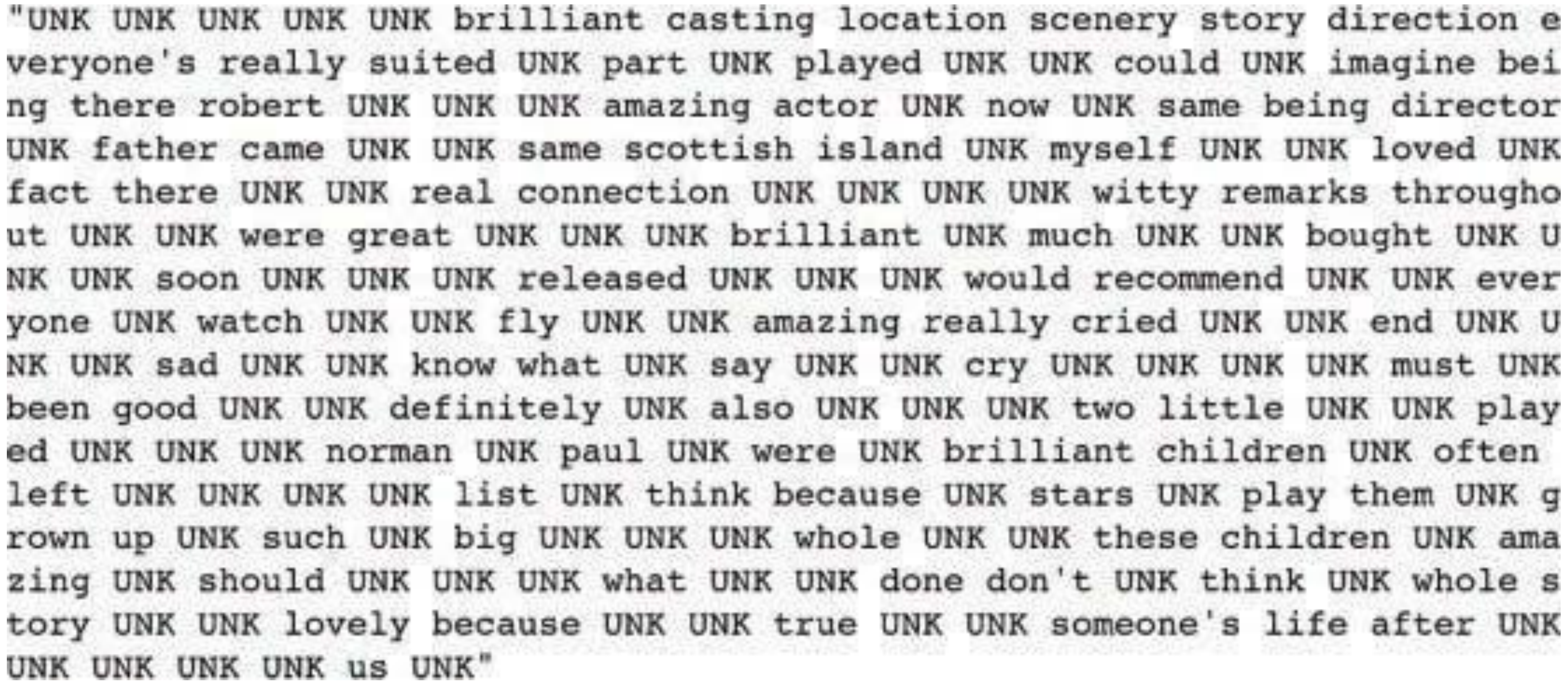
EXAMINING THE IMDB DATA

- ✓Executing x_train[0:6], shows the first six reviews from the training dataset
- ✓These reviews are natively in an integer-index format, where each unique token from the dataset is represented by an integer.
- ✓The first few integers are special cases, following a general convention that is widely used in NLP:
  - ✓0: Reserved as the padding token (which we'll soon add to the reviews that are shorter than max_review_length).
  - ✓1: Would be the starting token, which would indicate the beginning of a review. As per the next bullet point, however, the starting token is among the top 50 most common tokens and so is shown as "unknown."
  - ✓2: Any tokens that occur very frequently across the corpus (i.e., they're in the top 50 most common words) or rarely (i.e., they're below the top 5,050 most common words) will be outside of our word-vector vocabulary and so are replaced with this unknown token.
  - ✓3: The most frequently occurring word in the corpus.
  - ✓4: The second-most frequently occurring word.
  - ✓5: The third-most frequently occurring, and so on.
- ✓Looking at the length, we see that they are variable, ranging from 43 tokens up to 550 tokens.
- ✓The next step is to standardize all reviews to the same length.

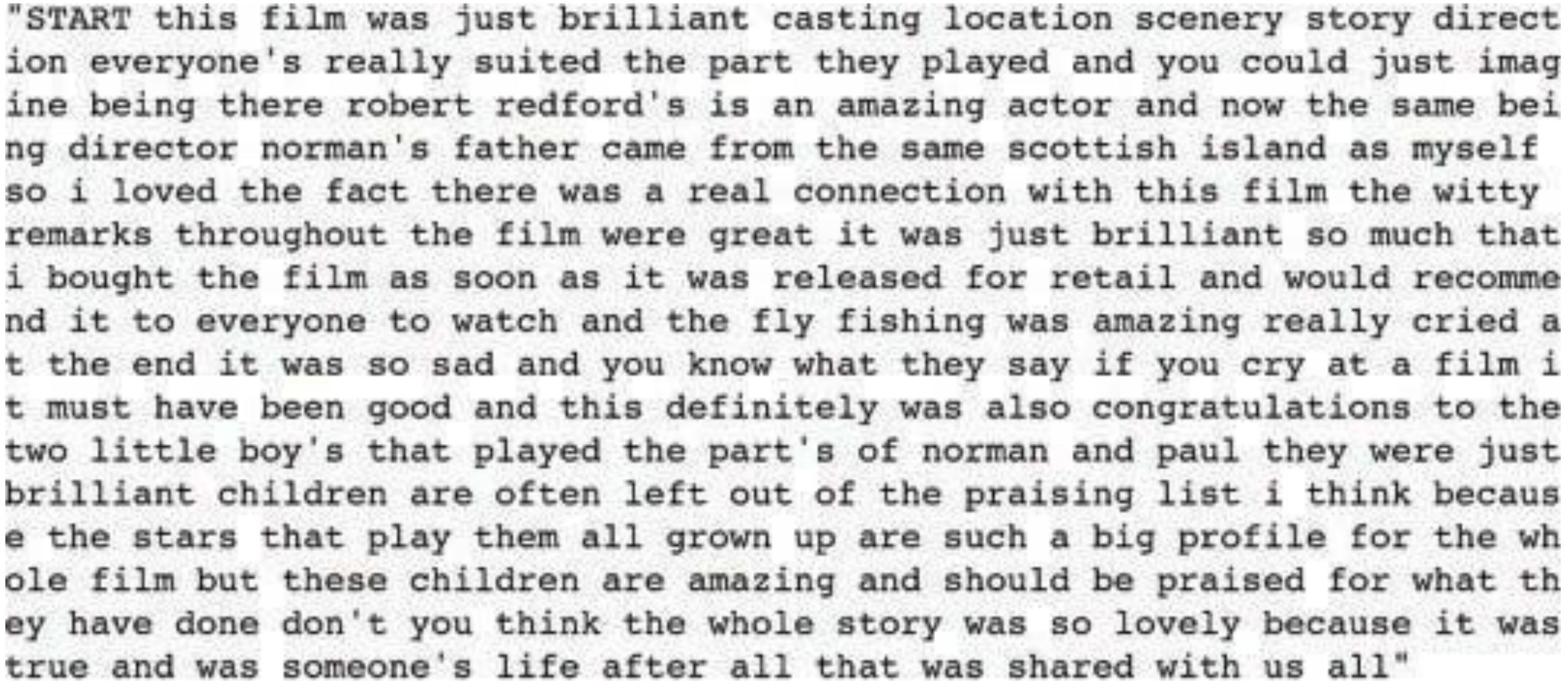
EXAMINING THE IMDB DATA

- ✓The film reviews are fed into the neural network model in the integer-index format because this is a memory-efficient way to store the token information.
- ✓It would require appreciably more memory to feed the tokens in as character strings.
- ✓For humans, however, it is uninformative (and, frankly, uninteresting) to examine reviews in the integer-index format.
- ✓To view the reviews as natural language, we create an index of words, where PAD, START, and UNK are customary for representing padding, starting, and unknown tokens, respectively.
- ✓It might be useful to have a mechanism to read the reviews without the UNK symboldebugging model results, it might indeed even
  - ✓For debugging results, it is practical to be able to view the full review.
  - ✓Brings out whether we are too aggressive or conservative with either our n_unique_words or n_words_to_skip thresholds
  - ✓Compare a full review with a truncated, pre-processed one

```
word_index = keras.datasets.imdb.get_word_index()
word_index = {k:(v+3) for k,v in word_index.items()}
word_index["PAD"] = 0
word_index["START"] = 1
word_index["UNK"] = 2
index_word = {v:k for k,v in word_index.items()}
```



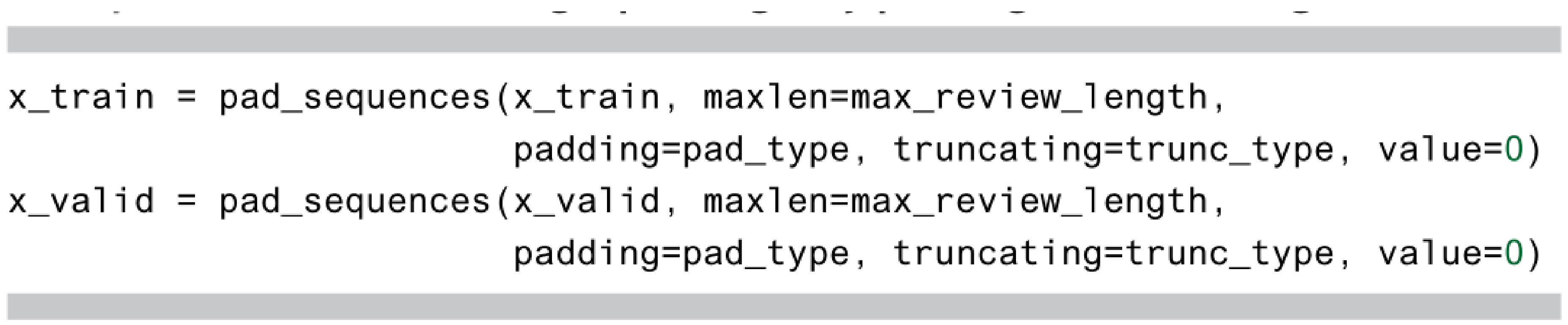
```
"UNK UNK UNK UNK UNK brilliant casting location scenery story direction e
veryone's really suited UNK part UNK played UNK UNK could UNK imagine bei
ng there robert UNK UNK UNK amazing actor UNK now UNK same being director
UNK father came UNK UNK same scottish island UNK myself UNK UNK loved UNK
fact there UNK UNK real connection UNK UNK UNK UNK witty remarks througho
ut UNK UNK were great UNK UNK UNK brilliant UNK much UNK UNK bought UNK U
NK UNK soon UNK UNK UNK released UNK UNK UNK would recommend UNK UNK ever
yone UNK watch UNK UNK fly UNK UNK amazing really cried UNK UNK end UNK U
NK UNK sad UNK UNK know what UNK say UNK UNK cry UNK UNK UNK UNK must UNK
been good UNK UNK definitely UNK also UNK UNK UNK two little UNK UNK play
ed UNK UNK UNK norman UNK paul UNK were UNK brilliant children UNK often
left UNK UNK UNK UNK list UNK think because UNK stars UNK play them UNK g
rown up UNK such UNK big UNK UNK UNK whole UNK UNK these children UNK ama
zing UNK should UNK UNK UNK what UNK UNK done don't UNK think UNK whole s
tory UNK UNK lovely because UNK UNK true UNK UNK someone's life after UNK
UNK UNK UNK UNK us UNK"
```



```
"START this film was just brilliant casting location scenery story direct
ion everyone's really suited the part they played and you could just imag
ine being there robert redford's is an amazing actor and now the same bei
ng director norman's father came from the same scottish island as myself
so i loved the fact there was a real connection with this film the witty
remarks throughout the film were great it was just brilliant so much that
i bought the film as soon as it was released for retail and would recomme
nd it to everyone to watch and the fly fishing was amazing really cried a
t the end it was so sad and you know what they say if you cry at a film i
t must have been good and this definitely was also congratulations to the
two little boy's that played the part's of norman and paul they were just
brilliant children are often left out of the praising list i think becaus
e the stars that play them all grown up are such a big profile for the wh
ole film but these children are amazing and should be praised for what th
ey have done don't you think the whole story was so lovely because it was
true and was someone's life after all that was shared with us all"
```

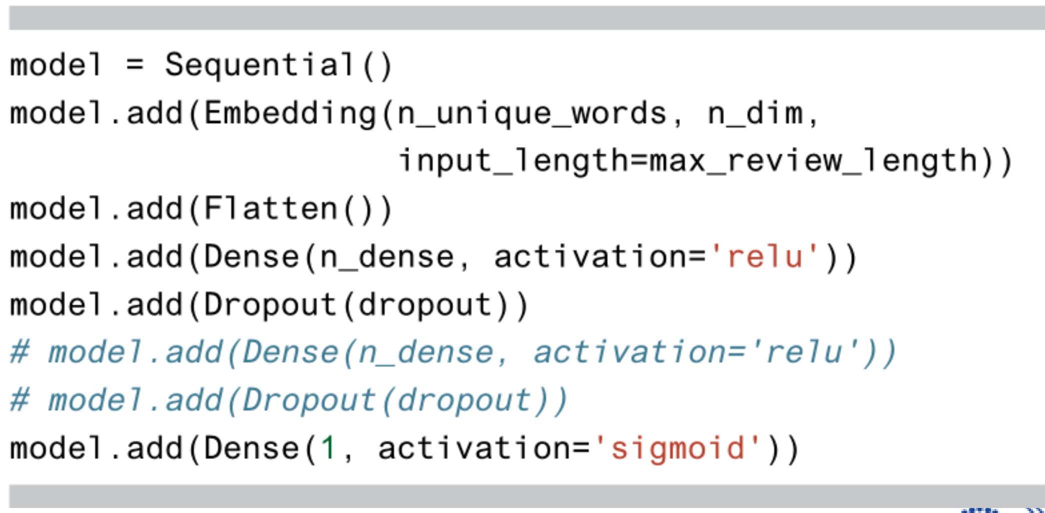
STANDARDIZING THE LENGTH OF THE REVIEWS

- In the current dataset there is variability in the length of the film reviews.
- The Keras-created TensorFlow model needs to know the size of the inputs that will be flowing into the model during training.
- This enables TensorFlow to optimize the allocation of memory and compute resources.
- Keras provides a convenient pad_sequences() method that enables us to both pad and truncate documents of text in a single line.



```
x_train = pad_sequences(x_train, maxlen=max_review_length,
                        padding=pad_type, truncating=trunc_type, value=0)
x_valid = pad_sequences(x_valid, maxlen=max_review_length,
                        padding=pad_type, truncating=trunc_type, value=0)
```

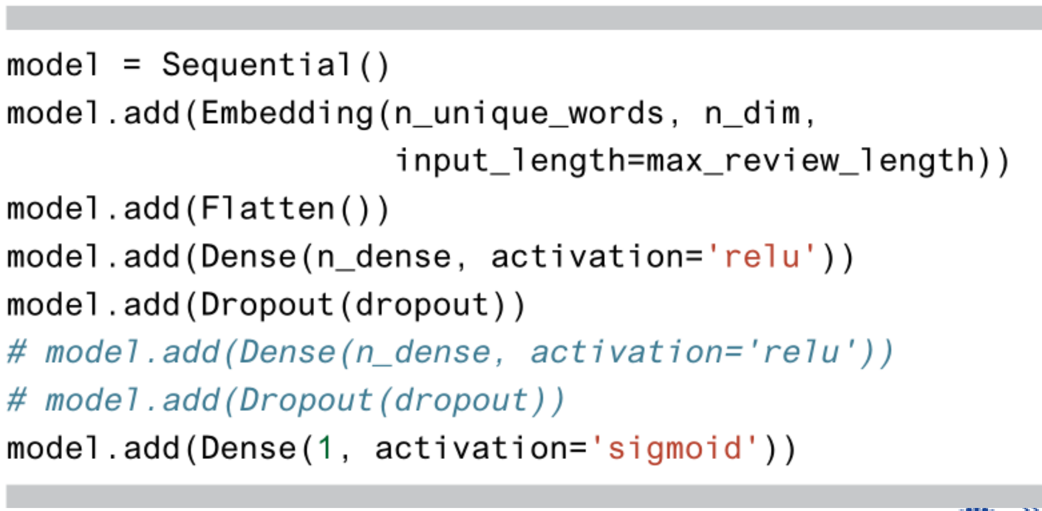
DENSE SENTIMENT CLASSIFIER ARCHITECTURE



```
model = Sequential()
model.add(Embedding(n_unique_words, n_dim,
                    input_length=max_review_length))
model.add(Flatten())
model.add(Dense(n_dense, activation='relu'))
model.add(Dropout(dropout))
# model.add(Dense(n_dense, activation='relu'))
# model.add(Dropout(dropout))
model.add(Dense(1, activation='sigmoid'))
```

- A Keras Sequential() method is used to invoke a sequential model
- The Embedding() layer enables us to create word vectors from a corpus of documents, i.e. 25,000 movie reviews
- Relative to independently creating word vectors with word2vec (or GloVe, etc.), training your word vectors via backpropagation as a component of your broader NLP model has a potential advantage:
- The locations that words are assigned to within the vector space reflect both the word similarity and the relevance of the words to the ultimate, specific purpose of the model (e.g., binary classification of IMDb reviews by sentiment).
- The size of the word-vector vocabulary and the number of dimensions of the vector space are specified by **n_unique_words** and **n_dim**, respectively.
- We pass the shape of the input layer to the first hidden layer in the network – the embedding layer using the the input_length argument.

DENSE SENTIMENT CLASSIFIER ARCHITECTURE

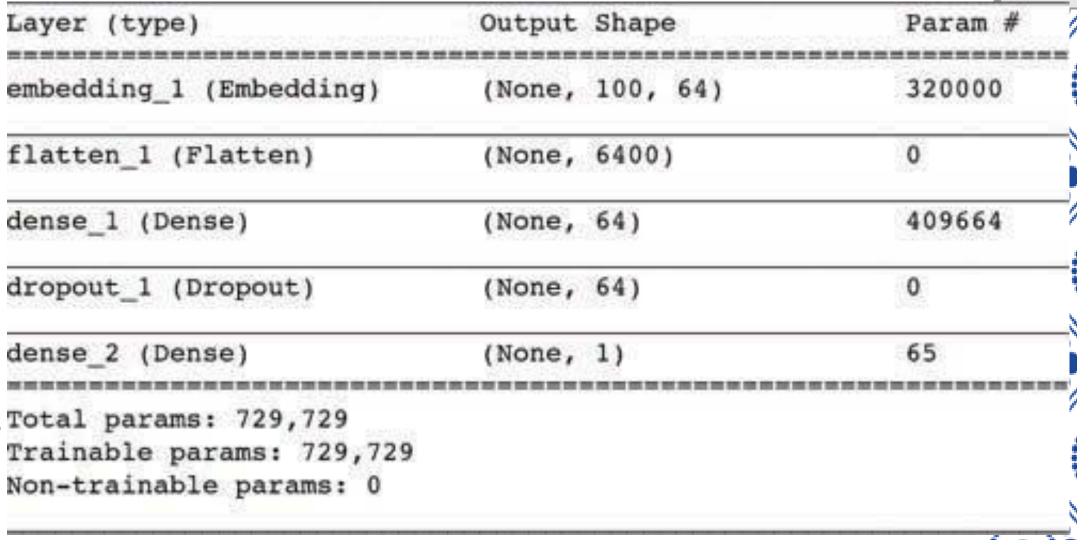


```
model = Sequential()
model.add(Embedding(n_unique_words, n_dim,
                    input_length=max_review_length))
model.add(Flatten())
model.add(Dense(n_dense, activation='relu'))
model.add(Dropout(dropout))
# model.add(Dense(n_dense, activation='relu'))
# model.add(Dropout(dropout))
model.add(Dense(1, activation='sigmoid'))
```

- ✓The Flatten() layer enables us to pass a many-dimensional output (here, a two-dimensional output from the embedding layer) into a one-dimensional dense layer.
- ✓A single Dense layer with Dropout() applied to it is used.
- ✓The architecture is a fairly shallow neural network architecture
- ✓This can be trivially deepened by adding further Dense() layers
- ✓Finally, because there are only two classes to classify, we require only a single output neuron
  - ✓if one class has the probability p then the other class has the probability 1 − p)
  - ✓This neuron is sigmoid for the output probabilities to be in the range between 0 and 1

MODEL SUMMARY

- The fairly simple model has quite a bit of parameters
- The embedding layer has 320,000 parameters
  - They come from having 5,000 words, each one with a location specified in a 64-dimensional word-vector space (64 × 5,000 = 320,000).



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 100, 64)           320000
_________________________________________________________________
flatten_1 (Flatten)          (None, 6400)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                409664
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 729,729
Trainable params: 729,729
Non-trainable params: 0
_________________________________________________________________
```

Flowing out of the embedding layer through the flatten layer and into the dense layer are 6,400 values:

❖Each of our film-review inputs consists of 100 tokens, with each token specified by 64 word-vector-space coordinates (64 × 100 = 6,400).

Each of the 64 neurons in the dense hidden layer receives input from each of the 6,400 values flowing out of the flatten layer, for a total of 64 × 6,400 = 409,600 weights.

Each of the 64 neurons has a bias, for a total of 409,664 parameters in the layer.

Finally, the single neuron of the output layer has 64 weights—one for the activation output by each of the neurons in the preceding layer—plus its bias, for a total of 65 parameters.

Summing up the parameters from each of the layers, we have a grand total of 729,729 of them.

RECURRENT NEURAL NETWORKS

NEURAL NETWORKS DESIGNED FOR SEQUENTIAL DATA

RNNs – Recurrent Neural Networks

“Jon and Grant are writing a book together. They have really enjoyed writing it.”

Although the human mind can track the concepts in the second sentence quite easily, it is not easy for a neural network to do the same.

The convolutional sentiment classifier considers a word only in the context of two words on either side of it (k_conv = 3)

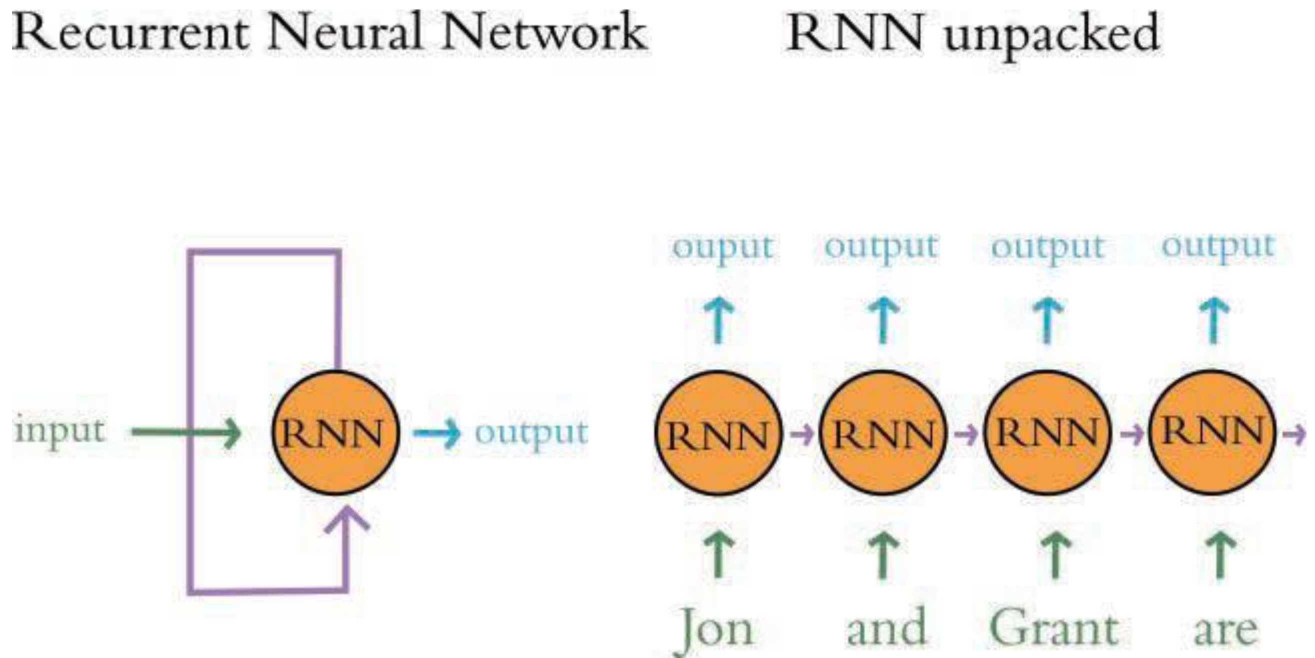
This limitation does not allow the neural network to assess what “they” or “it” might be referring to.

NEURAL NETWORKS DESIGNED FOR SEQUENTIAL DATA

- Human brains are able to keep track of objects from multiple sentences because human thoughts loop around each other.
- Humans can revisit earlier ideas in order to inform our understanding of the current context.
- Recurrent neural networks or RNNs allow us to keep track of objects over time.
- **RNNs have loops built into their structure that allow information to persist over time**

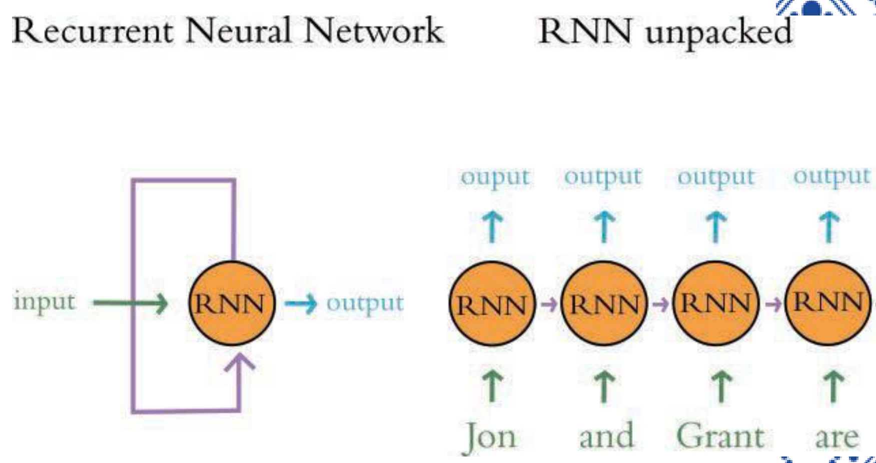
RECURRENT NEURAL NETWORKS

- The purple line indicates the loop that passes information between steps in the network.
- Similar to the dense network, there is a neuron for each input.
- There is a recurrent module for each word in the sentence.
- Each module receives an additional input from the previous module
- This allows the network to pass along information from earlier timesteps in the sequence.
- Recurrent neural networks are, computationally, more complex to train than exclusively "feedforward" neural networks like the dense nets and CNNs



RECURRENT NEURAL NETWORKS

- ✓Feedforward networks involve backpropagating cost from the output layer back toward the input layer.
- ✓For a recurrent layer (such as SimpleRNN, LSTM, or GRU) the cost must be backpropagated
  - ✓back toward the input layer
  - ✓back over the timesteps of the recurrent layer (from later timesteps back toward earlier timesteps).
- ✓The gradient of learning vanishes as we backpropagate the weights over later hidden layers toward earlier ones
- ✓Similarly, the gradient also vanishes as we backpropagate over later timesteps within a recurrent layer toward earlier ones.
- ✓This causes **later timesteps in a sequence to have more influence within the model than earlier timesteps**.



TYPICAL CODING STEPS

- Loading dependencies
- **Set hyperparameters**
- Loading IMDb film review data
- Standardizing review length
- **Design the neural network architecture**
- Compiling the model
- Creating the ModelCheckpoint() object and directory
- Fitting the model
- Loading the model parameters from the best epoch, with the critical exception that the particular epoch we select to load varies depending on which epoch has the lowest validation loss.
- Predicting ˆy for all validation data
- Plotting a histogram of ˆy
- Calculating ROC AUC

CHANGES IN RNN CODE

- Overfitting does not occur early in training, so the number of epochs was quadrupled to 16.
- The **max_review_length** reduced back down to 100 -- even this is excessive for a simple RNN.
- The model can backpropagate over about 10 timesteps before the gradient of learning vanishes completely.
- Thus, **max_review_length** could probably be lowered to less than 10 before a reduction in this model's performance is noticed.
- The word-vector vocabulary was set to 10000 tokens and this seemed to provide improved results for these RNN architectures.
- Setting n_rnn = 256 made the number of units in the recurrent layer as 256 units, or, 256 cells.
- 256 convolutional filters enabled the CNN model to specialize in detecting 256 unique triplets of word meaning
- This setting enables the RNN to detect 256 unique sequences of word meaning that may be relevant to review sentiment.

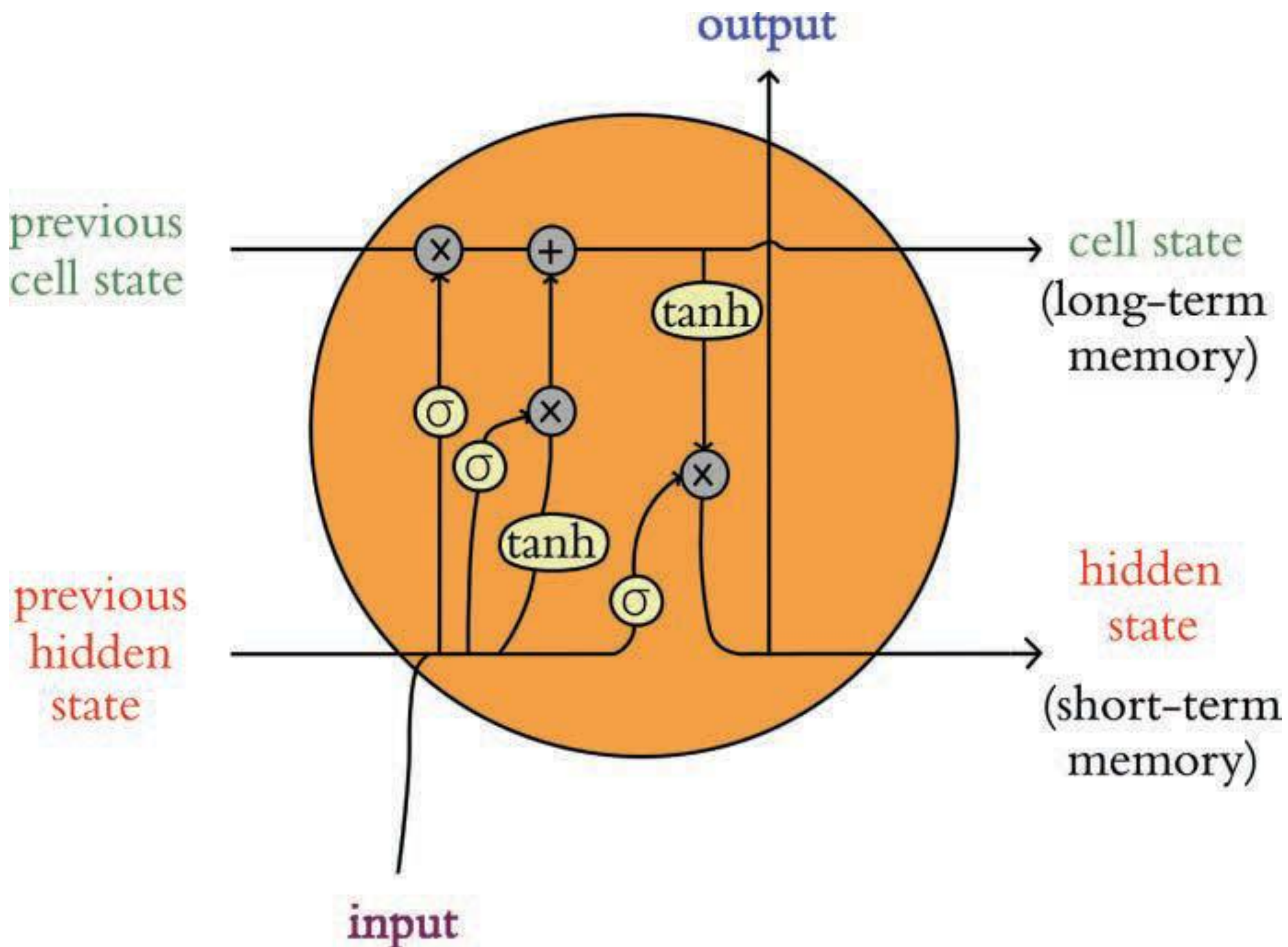
RESULTS OF RUNNING THE RNN CODE

- ✓The results of running this model i.e. the RNN Sentiment Classifier notebook were not encouraging.
- ✓The training loss starts jumping around after going down steadily over the first half-dozen steps.
- ✓Consequence of the model struggling to learn patterns even within the training data.
  - ✓Relative to the validation data the model should be readily able to learn patterns in the training data
- ✓For all other models, the training losses that reliably attenuated epoch over epoch.
- ✓The validation loss bounced around, along with the training loss.
- ✓The lowest validation loss is in the seventh epoch (0.504) and that corresponds to a validation accuracy of 77.6 percent and an ROC AUC of 84.9 percent.
- ✓All three of these metrics are our worst yet for a sentiment classifier model.
- ✓This is because RNNs are only able to backpropagate through ~10 time steps before the gradient diminishes so much that parameter updates become negligibly small.
- ✓Because of this, simple RNNs are rarely used in practice: More-sophisticated recurrent layer types like LSTMs, which can backpropagate through ~100 time steps, are far more common

LSTM UNITS

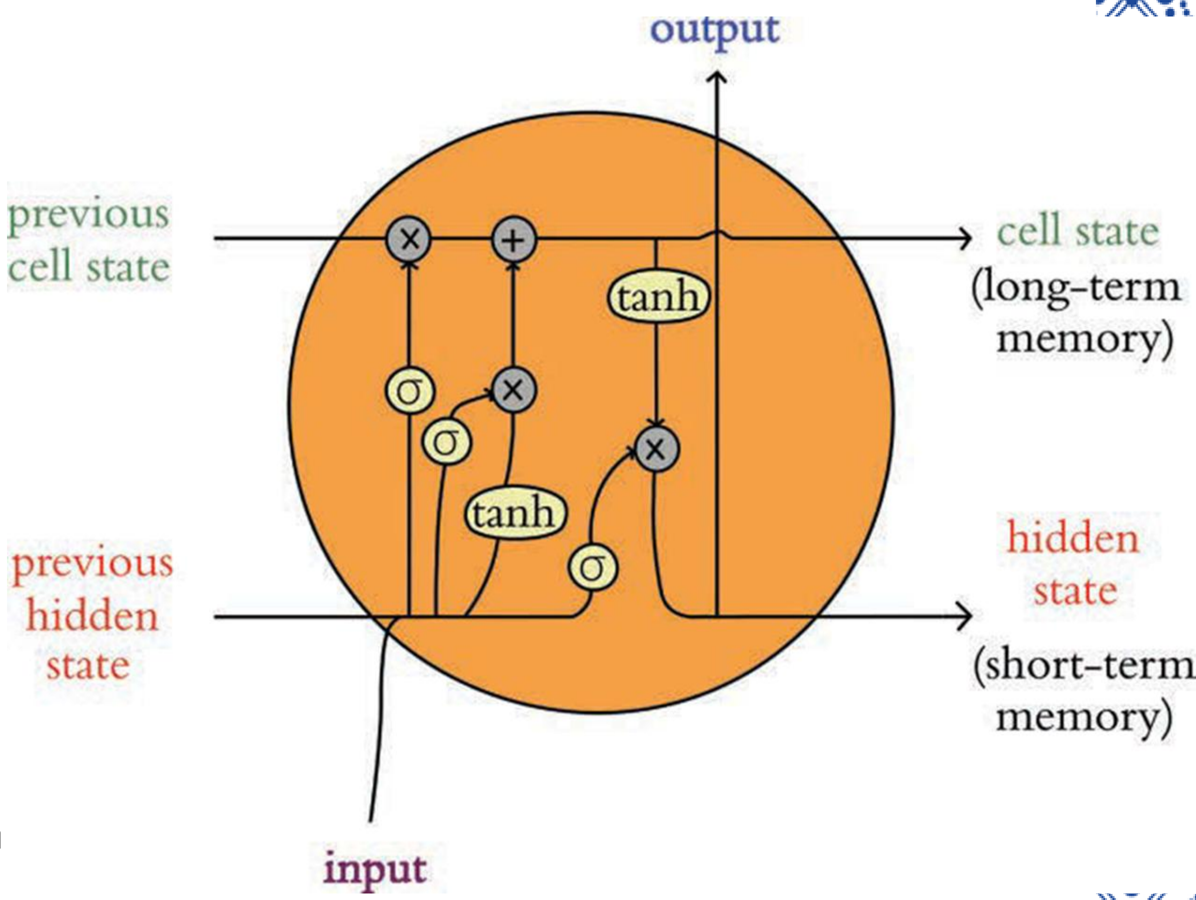
- ✓LSTMs were introduced by Sepp Hochreiter and Jürgen Schmidhuber in 1997
- ✓They are more widely used in NLP deep learning applications today than ever before.
- ✓The basic structure of an LSTM layer is the same as the simple recurrent layers
- ✓LSTMs receive input:
  - ✓from the sequence of data (e.g., a particular token from a natural language document), and
  - ✓they also receive input from the previous time point in the sequence.
- ✓The difference is that inside each cell in a simple recurrent layer (e.g., SimpleRNN() in Keras), we have a single neural network activation function such as a tanh function
- ✓This activation function transforms the RNN cell's inputs to generate its output.
- ✓In contrast, the cells of an LSTM layer contain a far more complex structure

LONG SHORT-TERM MEMORY (LSTM) UNITS



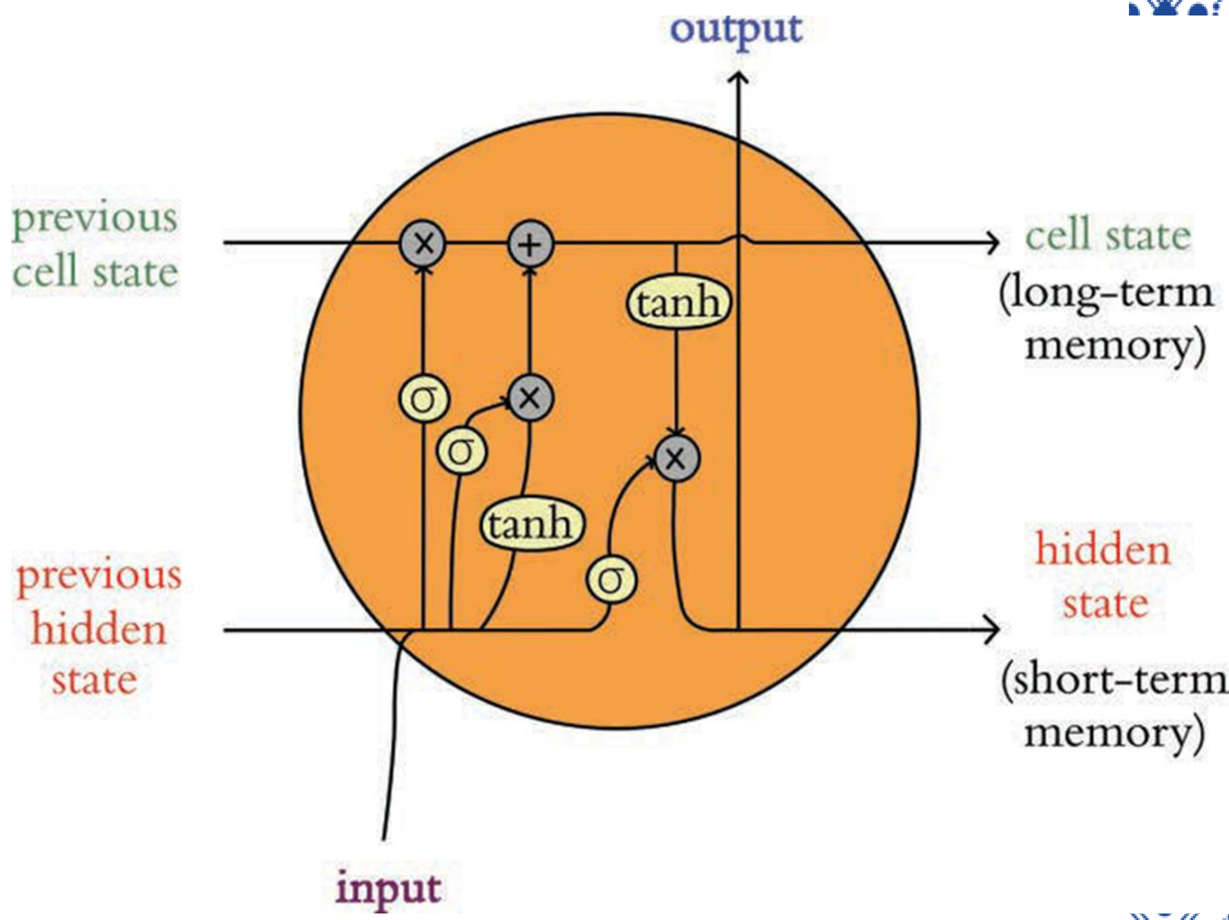
LSTM DETAILS [1]

- LSTM cell structure is quite complex
- The cell state runs across the top of the LSTM cell.
- The cell state does not pass through any nonlinear activation functions.
- It undergoes some minor linear transformations, but otherwise it simply passes through from cell to cell.
- Two linear transformations (a multiplication and an addition operation) are points where a cell in an LSTM layer can add information to the cell state.
- This information will be passed onto the next cell in the layer.
- In either case, there is a sigmoid activation (represented by σ in the figure) before the information is added to the cell state.
- These sigmoids act as gates because a sigmoid activation produces values between 0 and 1.
- In effect, these sigmoid "gates" decide whether new information (from the current timestep) is added to the cell state or not.



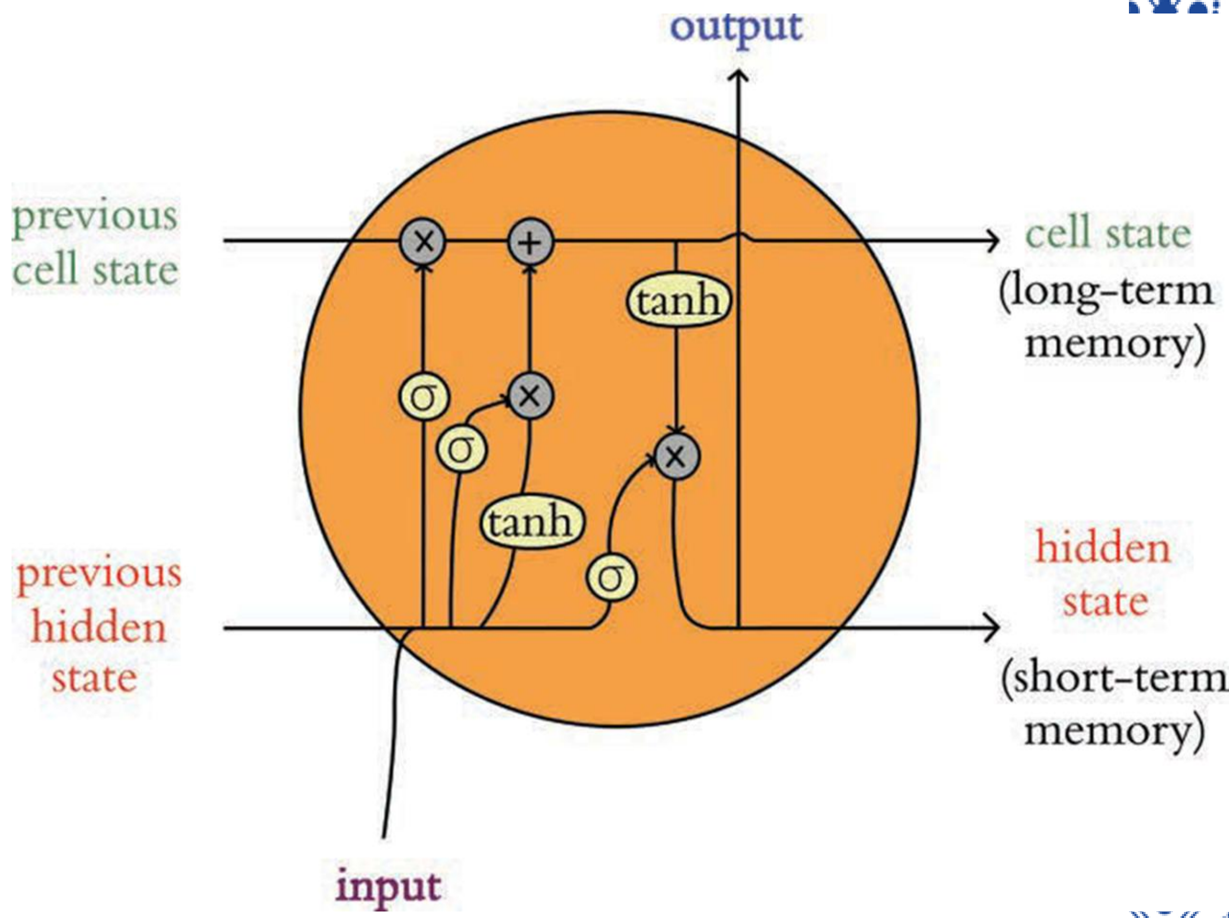
LSTM DETAILS [2]

- The new information at the current timestep is a simple concatenation of **the current timestep's input** and **the hidden state from the preceding timestep**.
- This concatenation has two chances to be incorporated into the cell state—either linearly or following a nonlinear tanh activation.
- In both cases, it's the sigmoid gates that decide whether the information is combined.



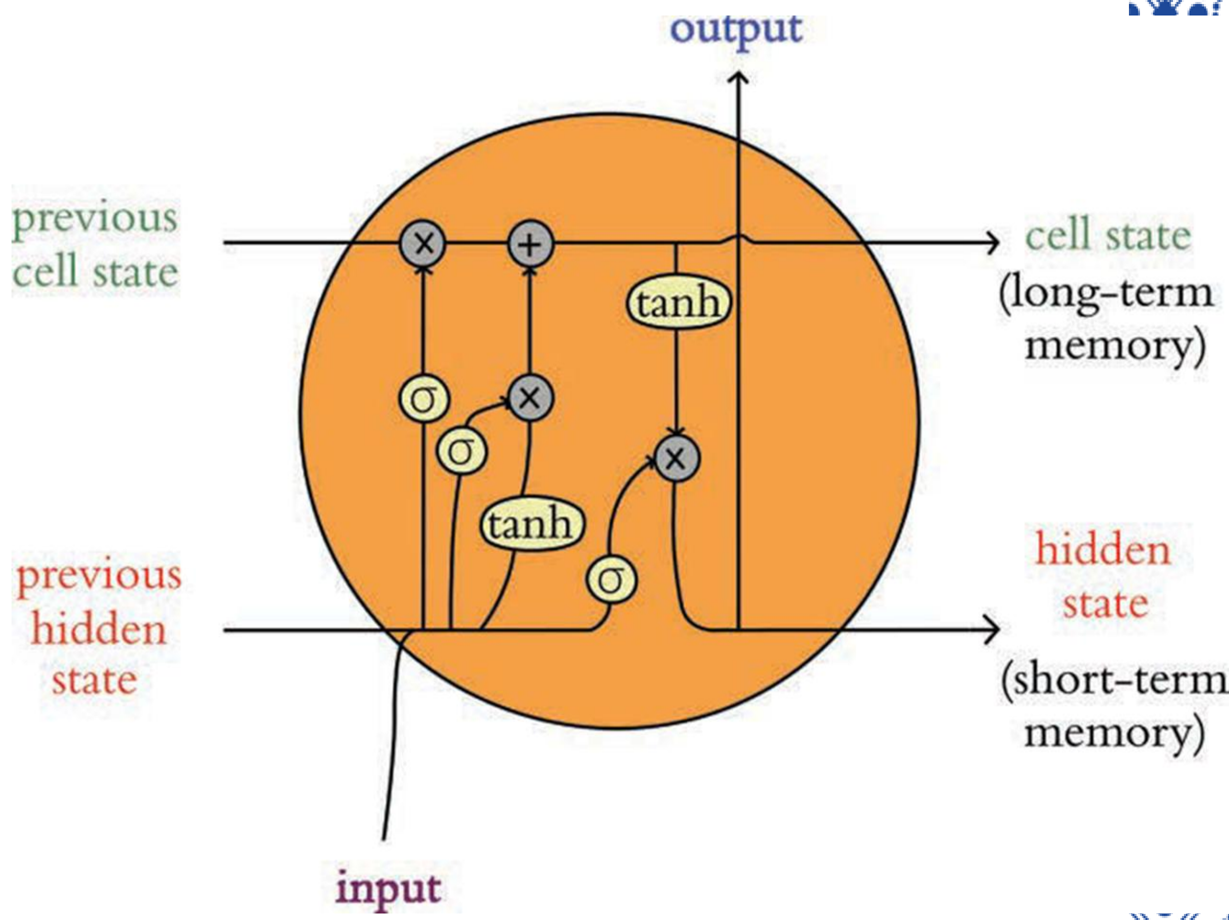
LSTM DETAILS [3]

- The LSTM first determines which information should be added to the cell state
- Then, another sigmoid gate decides whether the information from the current input is added to the final cell state, and this results in the output for the current timestep.
- Notice that, under a different name ("hidden state"), the output is also sent into the next LSTM module
- This represents the next timestep in the sequence
- Here it is combined with the next timestep's input to begin the whole process again
- The final cell state alongside the hidden state is sent to the module representing the next timestep.



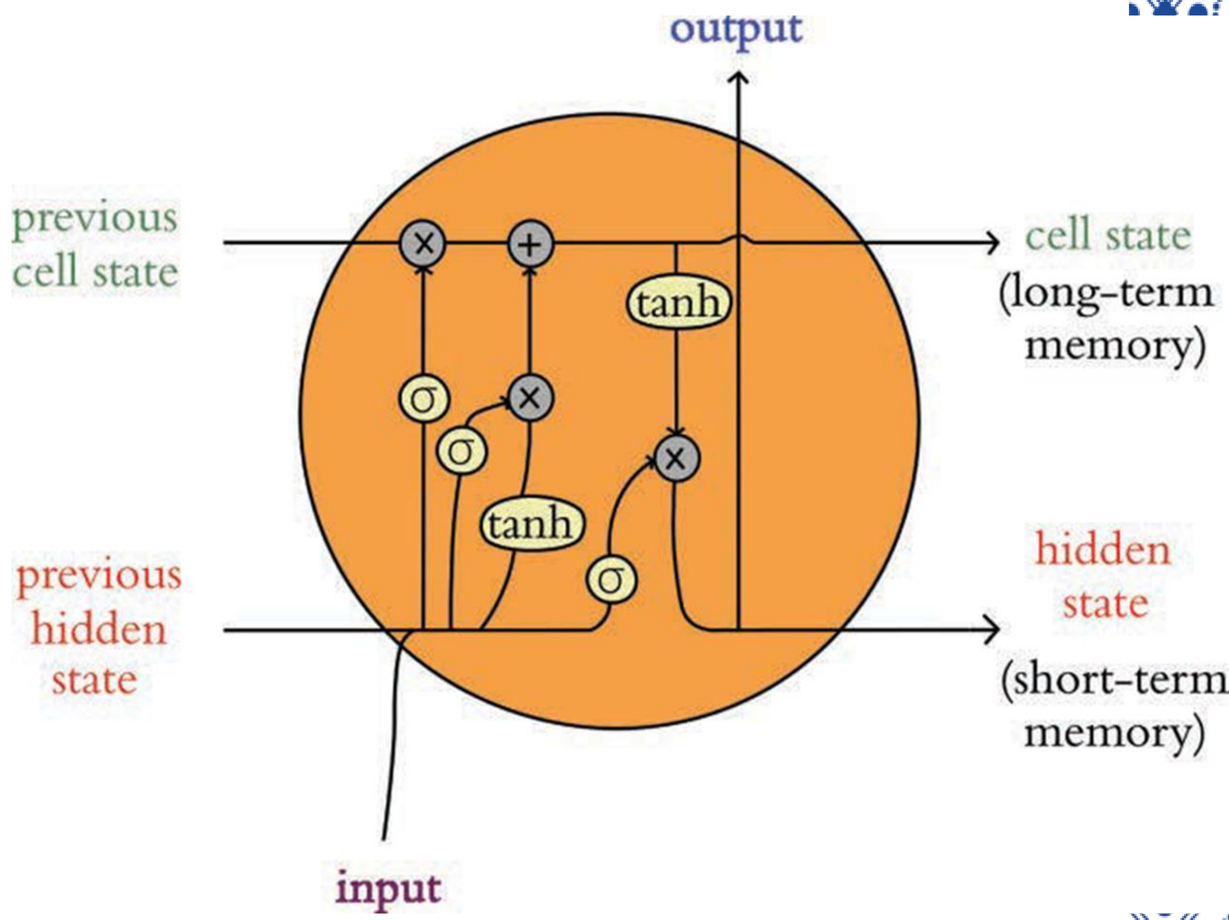
LSTM DETAILS [4]

- The cell state enables information to persist along the length of the sequence, through each timestep in a given LSTM cell. It is the long-term memory of the LSTM.
- The hidden state is analogous to the recurrent connections in a simple RNN and represents the short-term memory of the LSTM.
- Each module represents a particular point in the sequence of data (e.g., a particular token from a natural language document).
- At each timestep, several decisions are made (using those sigmoid gates) about whether the information at that particular timestep in the sequence is relevant to the local (hidden state) and global (cell state) contexts.



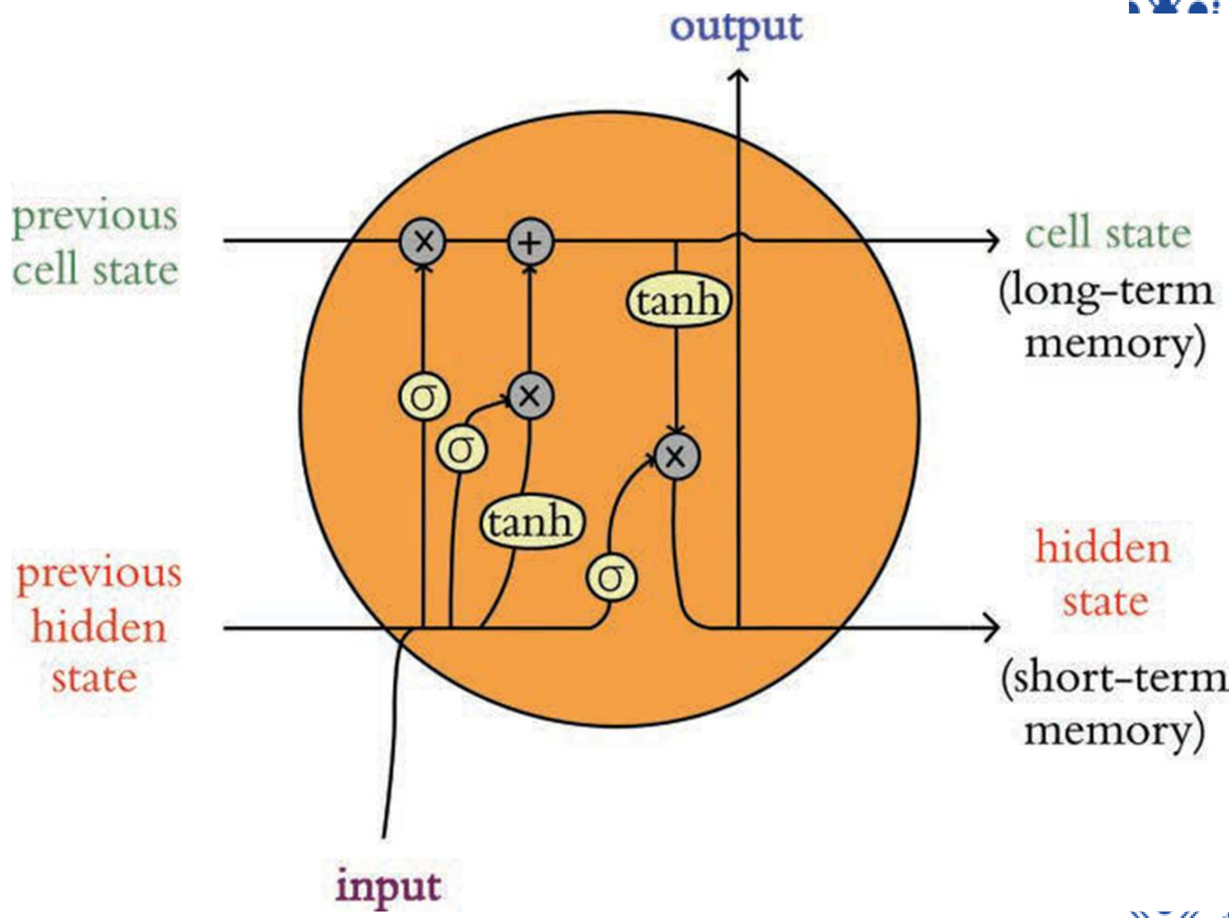
LSTM DETAILS [5]

- ✓The first two sigmoid gates determine:
  - ✓whether the information from the current timestep is relevant to the global context (the cell state) and
  - ✓how it will be combined into that stream.
- ✓The final sigmoid gate determines whether the information from the current timestep is relevant to the local context
- ✓In other words, it determines whether it is added to the hidden state which is also the output for the current timestep



LSTM DETAILS [6]

- ✓Simple RNN cells pass only one type of information (the hidden state) between timesteps and contain only one activation function.
- ✓LSTM cells are much more complex:
  - ✓They pass two types of information between timesteps (hidden state and cell state) and
  - ✓They contain five activation functions.



BIDIRECTIONAL LSTMS

- Bidirectional LSTMs (or Bi-LSTMs, for short) are a clever variation on standard LSTMs.
- LSTMs involve backpropagation in only one direction (typically backward over timesteps, such as from the end of a movie review toward the beginning)
- Bidirectional LSTMs involve backpropagation in both directions (backward and forward over timesteps) across some one-dimensional input.
- This extra backpropagation doubles computational complexity.
- However, Bi-LSTMs are highly accurate.
- Bi-LSTMs → a popular choice in modern NLP applications because their ability to learn patterns both before and after a given token within an input document facilitates **high-performing models**.

THANKS