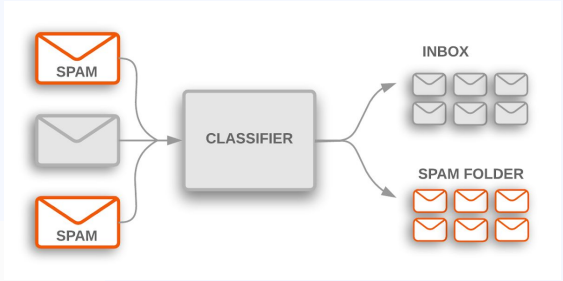
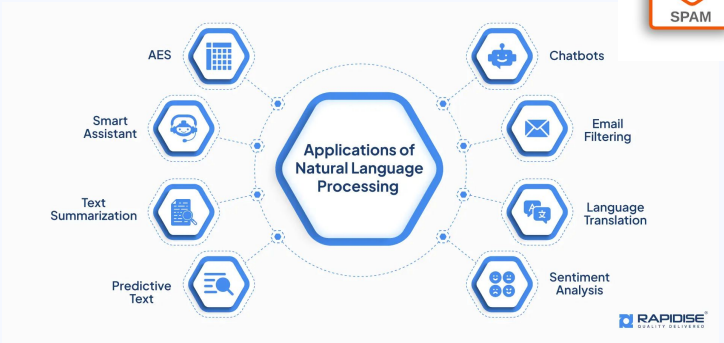
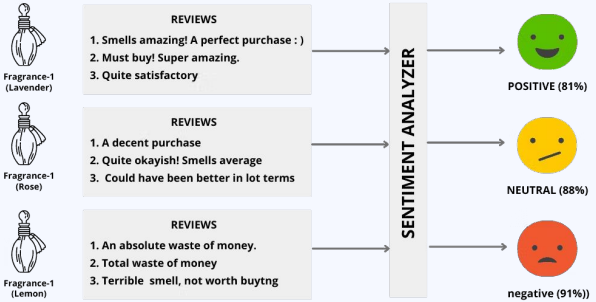
Natural Language Processing (NLP)

Natural Language Processing

- Why do computers need to process human language?





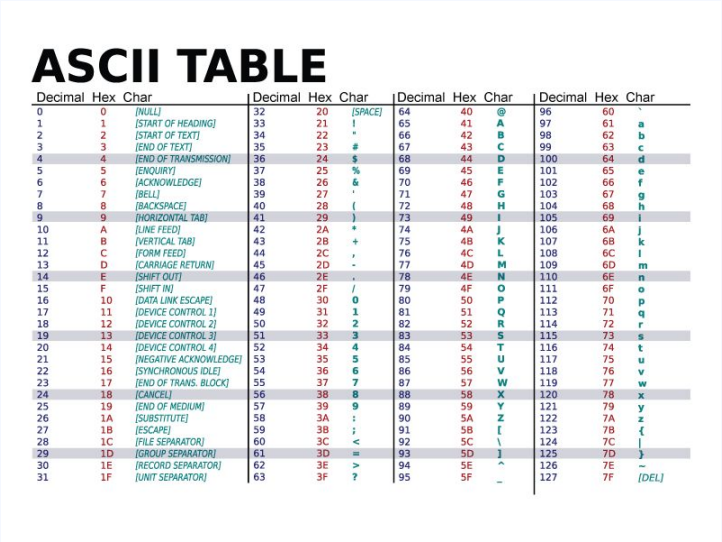


Natural Language Processing

- How do computers process text data?
- How to turn text into a bunch of numbers?

Natural Language Processing

- ASCII encoding, which turns characters into a set of numbers based on a lookup table
- Is this good enough for computers to actually gain an understanding of words, phrases, and sentences?

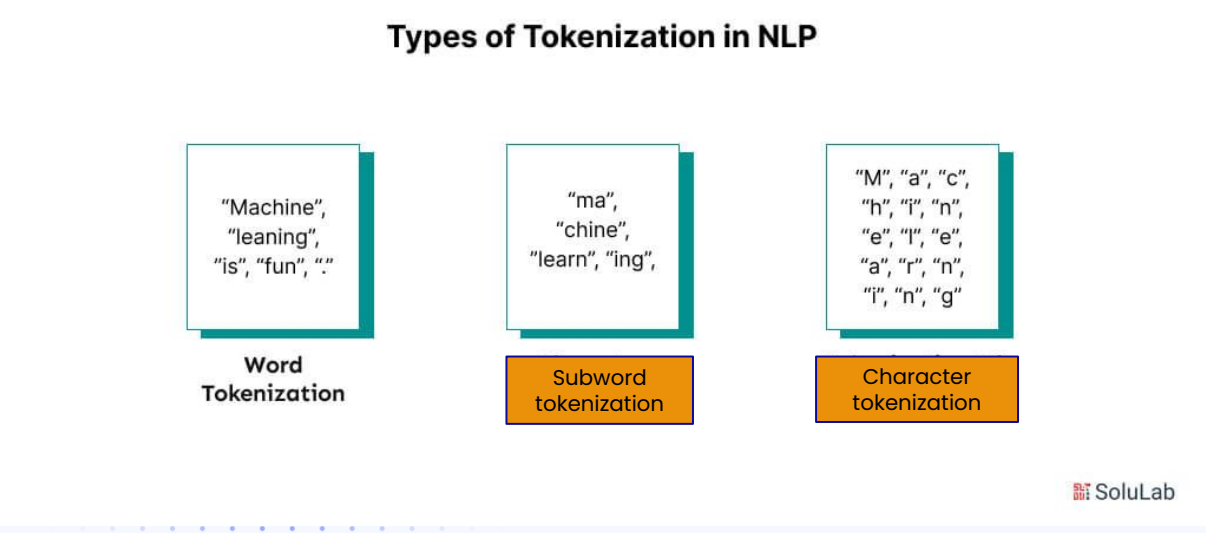


ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

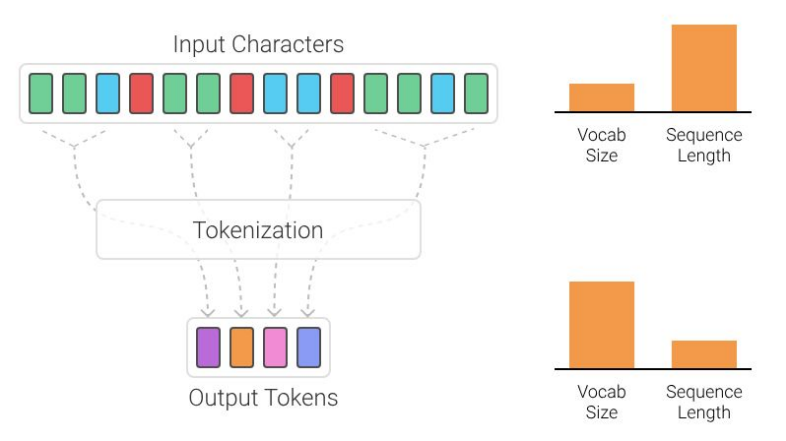
Tokenization

- There are many schemes for tokenization, but the most common scheme in state-of-the-art NLP systems involve splitting into words, word fragments, and punctuations.



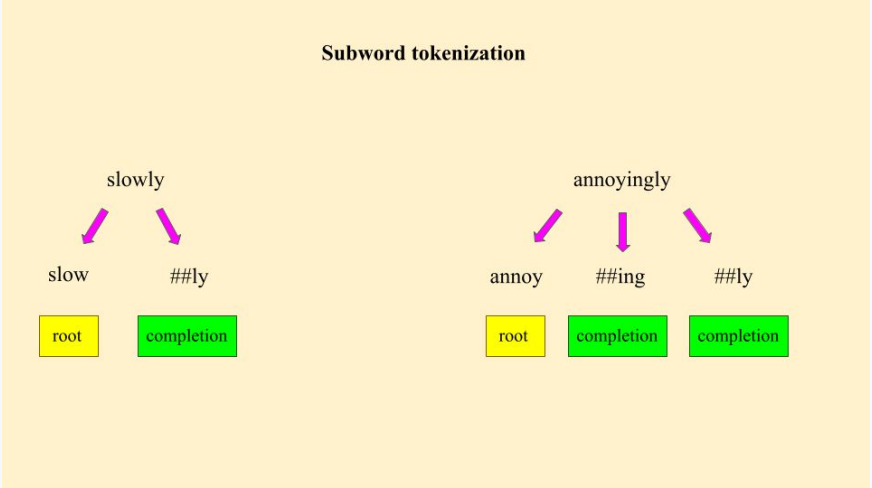
Tokenization

- Why words and word fragments (subword tokenization)?
- What are the alternative possible schemes?
- Character-based: hard to encode meaning of actual words
- Using words every time: too many distinct tokens to manage (as there are many variations, including typos, of a word)



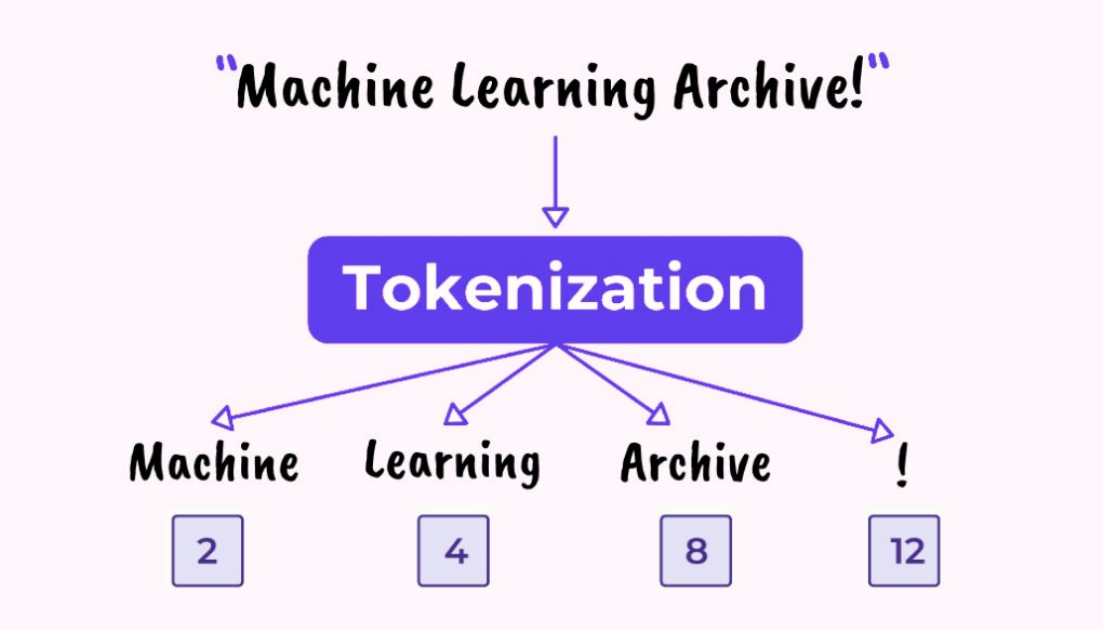
Tokenization

- Using subwords: captures the balance between preserving semantic meaning and making every token as distinct as possible
- The root words carry the main meaning, while the prefixes and suffixes carry information about how and where the word is used



Tokenization

- Each token in the sequence is mapped to an integer

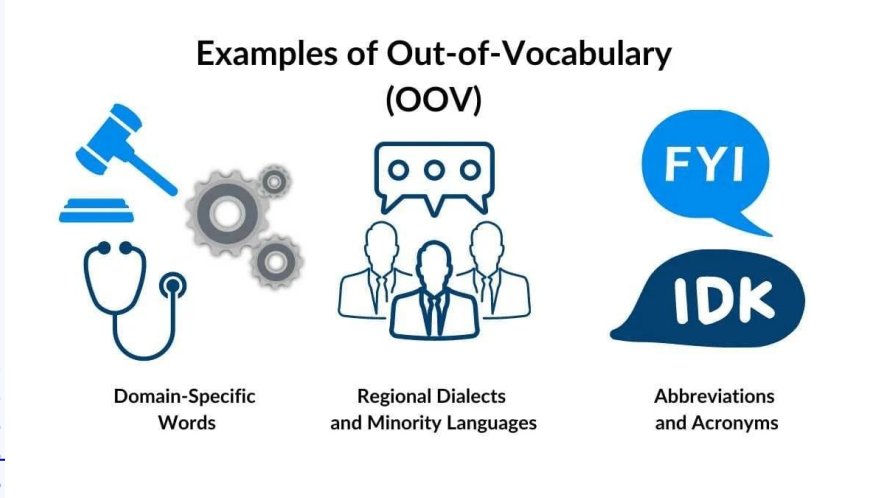


Vocabulary

- Tokenizers can only support a finite number of possible token options
- The set of possible tokens in a language model is called its vocabulary

Vocabulary

- Sometimes natural language offers us surprises by producing specific, rare words that are out of vocabulary (OOV) – they do not appear frequently enough in the training of the language model and so are not encoded into its tokenizer
- It's not easy to use traditional methods to handle these cases!

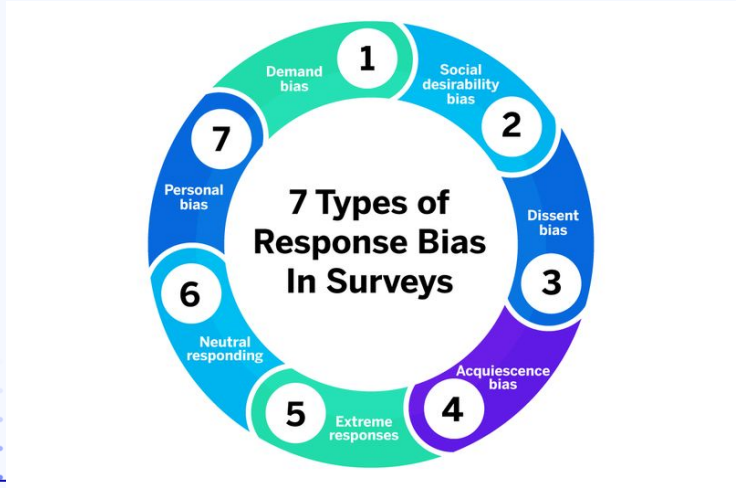


Data Quality in NLP

- Different problems in data quality can disrupt the operations of NLP systems
- Surface level problems:
- Typos and special characters
- Words with multiple distinct meanings (especially problematic for word embeddings)
- Code-switching (two languages appear at the same time, like what we're doing right now)
- Most modern LLMs (large language models) can now handle these cases smoothly, but they are worth noting if you use traditional methods

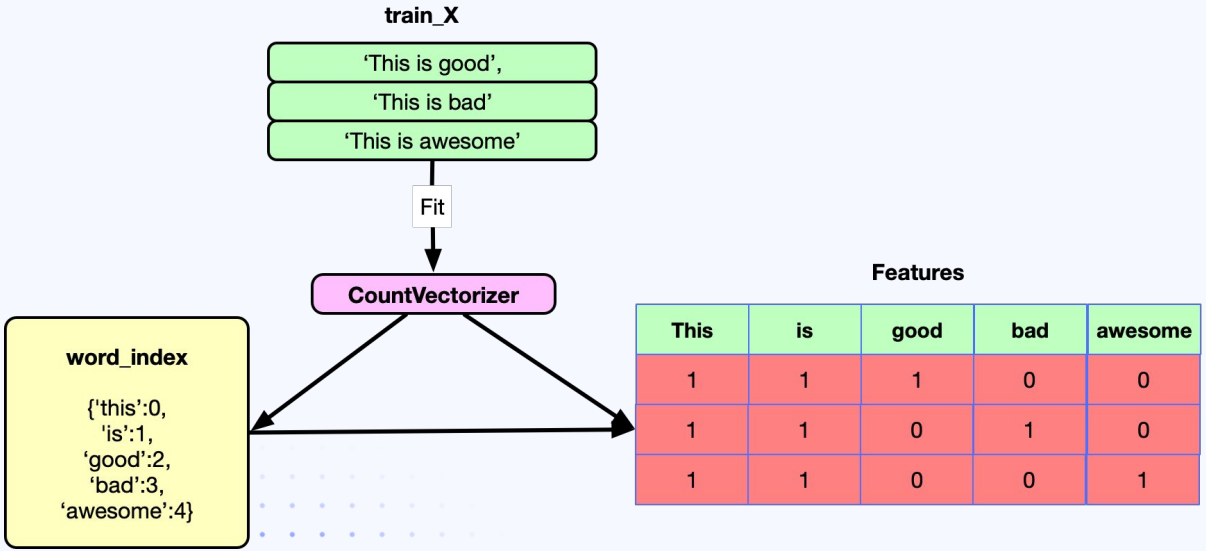
Data Quality in NLP

- Deeper problems:
- Ambiguities (like some exam questions where you need to carry out intent detection in order to get the mark)
- Biases in data (not only affects NLP, but computer vision and other stuff too)
- E.g., selection bias, response bias (the latter especially prevalent in surveys and reviews)



Encoding

- Text sequences can be encoded with information on the tokens it contains
- Convert an entire sequence into a single vector
- E.g., bag of words (e.g., CountVectorizer, TF-IDF)

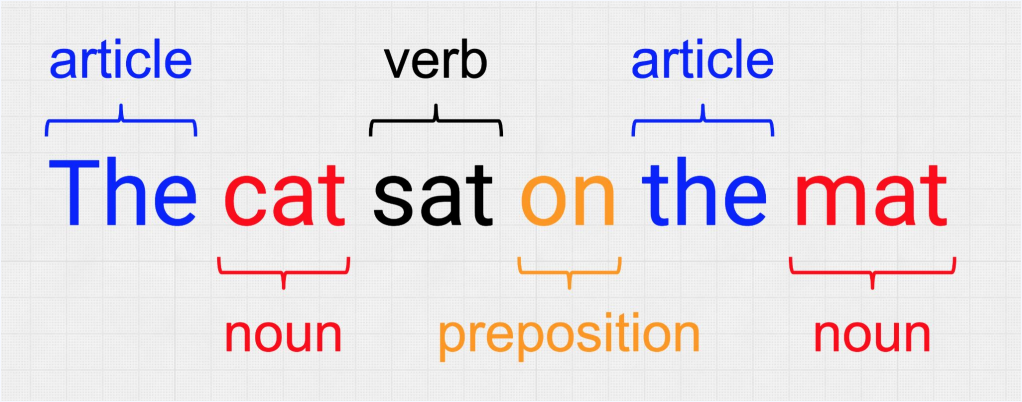


Encoding

- Bag of words is fitted on a text corpus to collect information about the words present in the training data
- Then transform a new sequence based on the words in the sequence that is also in the training corpus (i.e., no OOV handling)

Encoding

- Bag of words:
  Advantages: requires less computational resources, more interpretable
  Disadvantages: does not capture the ordering of tokens of the text sequence (e.g., cannot distinguish “the cat sat on the mat” from “the mat sat on the cat”)



Encoding

- Within the bag of words approach: TF-IDF is usually better than CountVectorizer
- Words like “a” and “the” commonly appear in sentences but do not convey much information; CountVectorizer amplifies these not-as-useful signals while TF-IDF masks them using inverse document frequency

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

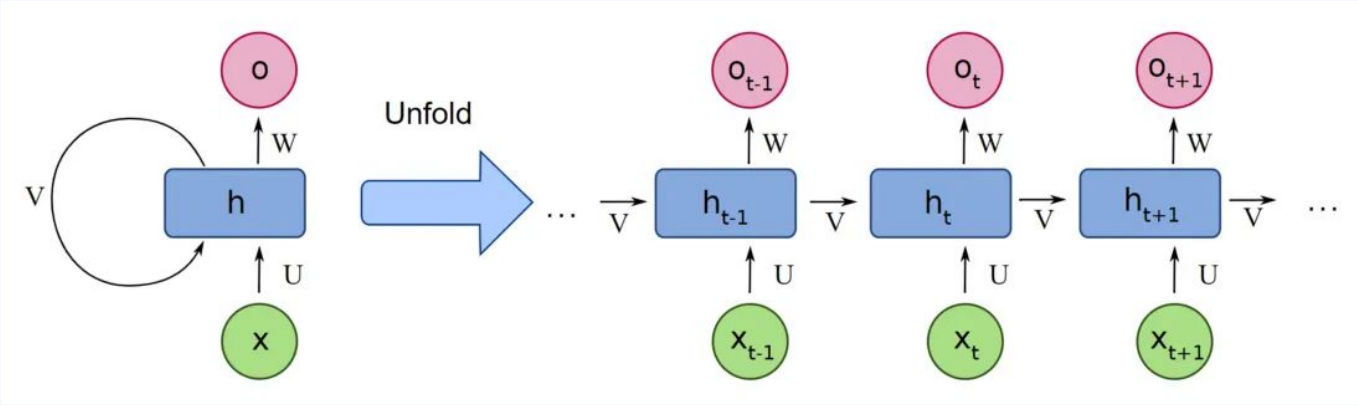
$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

Try it Out

- Try out this notebook using traditional NLP methods for sentiment analysis: https://www.kaggle.com/code/carsoncheng/review-sentiment

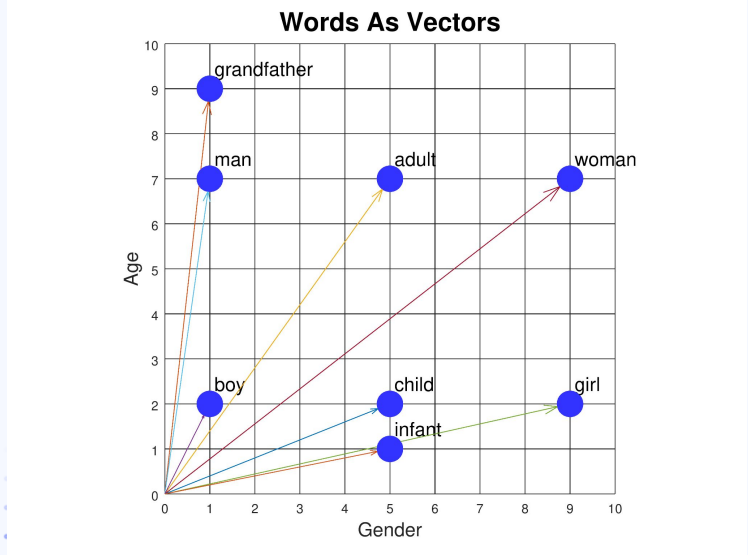
Sequence Modeling

- To model the order of tokens, sequence modeling methods are used
- This includes RNNs and LSTMs, which we will talk about in the time series slides



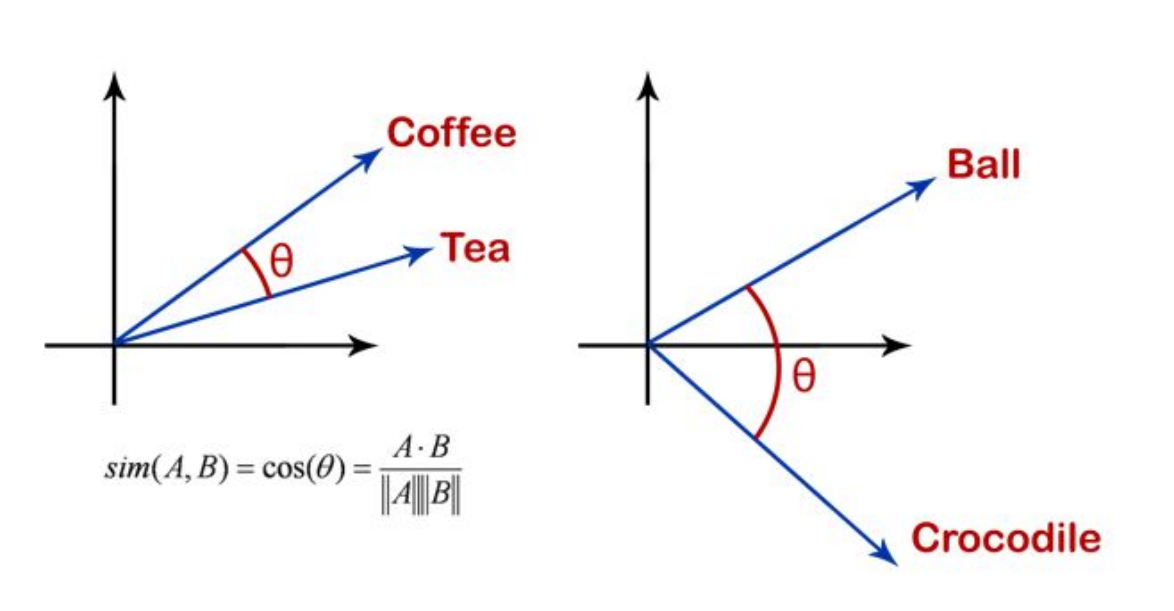
Embedding

- Map a token to a high-dimensional vector
- Based on semantic properties of the token; each dimension may correspond to a semantic property of the word



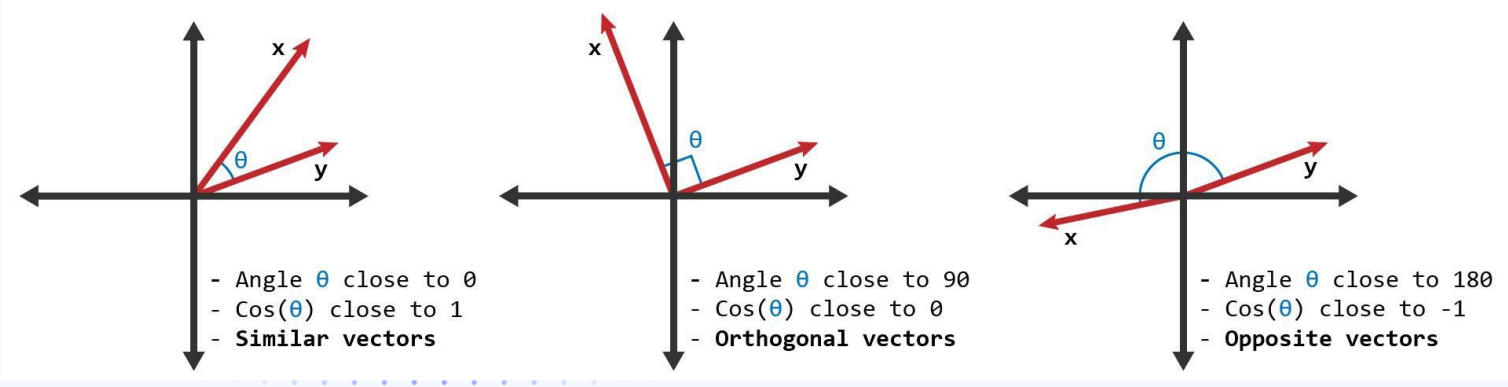
Embedding

- Tokens that are similar in meaning (or share semantic properties) get embeddings with high cosine similarity



Cosine Similarity

- Cosine similarity is a measure of similarity of two vectors between -1 and 1
- High cosine similarity (close to 1) between a pair of vectors mean they point to similar directions
- Cosine similarity close to 0 mean the two vectors are about perpendicular to each other
- Low cosine similarity (close to -1) between two vectors mean they point to opposite directions



Embedding

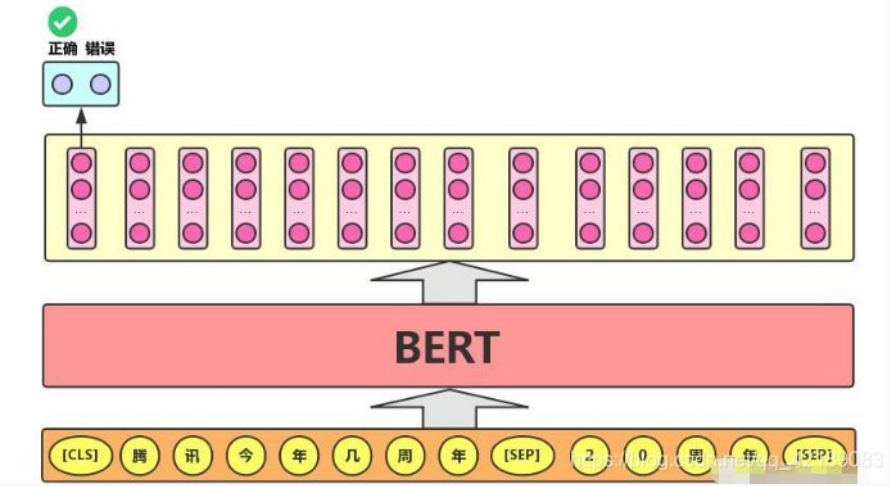
- Convert each token to its corresponding vector representation
- The task becomes modeling a sequence of vectors, which is done by language models

Try it Out

- Try out this notebook exploring the world of embeddings: https://www.kaggle.com/code/carsoncheng/glove-word-embeddings

Language Modeling

- NLP pipeline: text -> tokens -> embedding -> language modeling -> refined embeddings (both token-level and sequence-level) -> task-specific head



Language Modeling

- Common architectures:
- Encoder-only: BERT, SBERT (sentence transformers); they encode text into a latent representation
- Encoder-decoder: sequence-to-sequence, T5 models; they encode text into a latent representation, and then autoregressively decode off of it
- Decoder-only: GPT; they autoregressively generate text (one token at a time)
- (add some diagrams?)

Language Modeling

- Technically the embeddings (the vectors corresponding to each token) are also trainable, like other parameters in a language model; so the actual meaning of each dimension of the embedding may not be easy to interpret
- So the language model models a sequence of integers, which are the indices to the corresponding embeddings of tokens