UCL ARTIFICIAL
INTELLIGENCE SOCIETY

TUTORIAL #8

Session 8

Natural Language Processing

UCL ARTIFICIAL
INTELLIGENCE SOCIETY

LECTURE OVERVIEW

01

Intro to NLP

Why do we care about it?
Why it is so important?

02

Preprocessing

Preprocessing using NLTK (tokenization, stemming,lemmatization)

03

Embeddings

What are they? Popular methods for their creation,

04

Challenge

Use your NLP skills in practice!

UPDATES

Last competition

1. @ladams
2. @Ebart
3. @Mungo

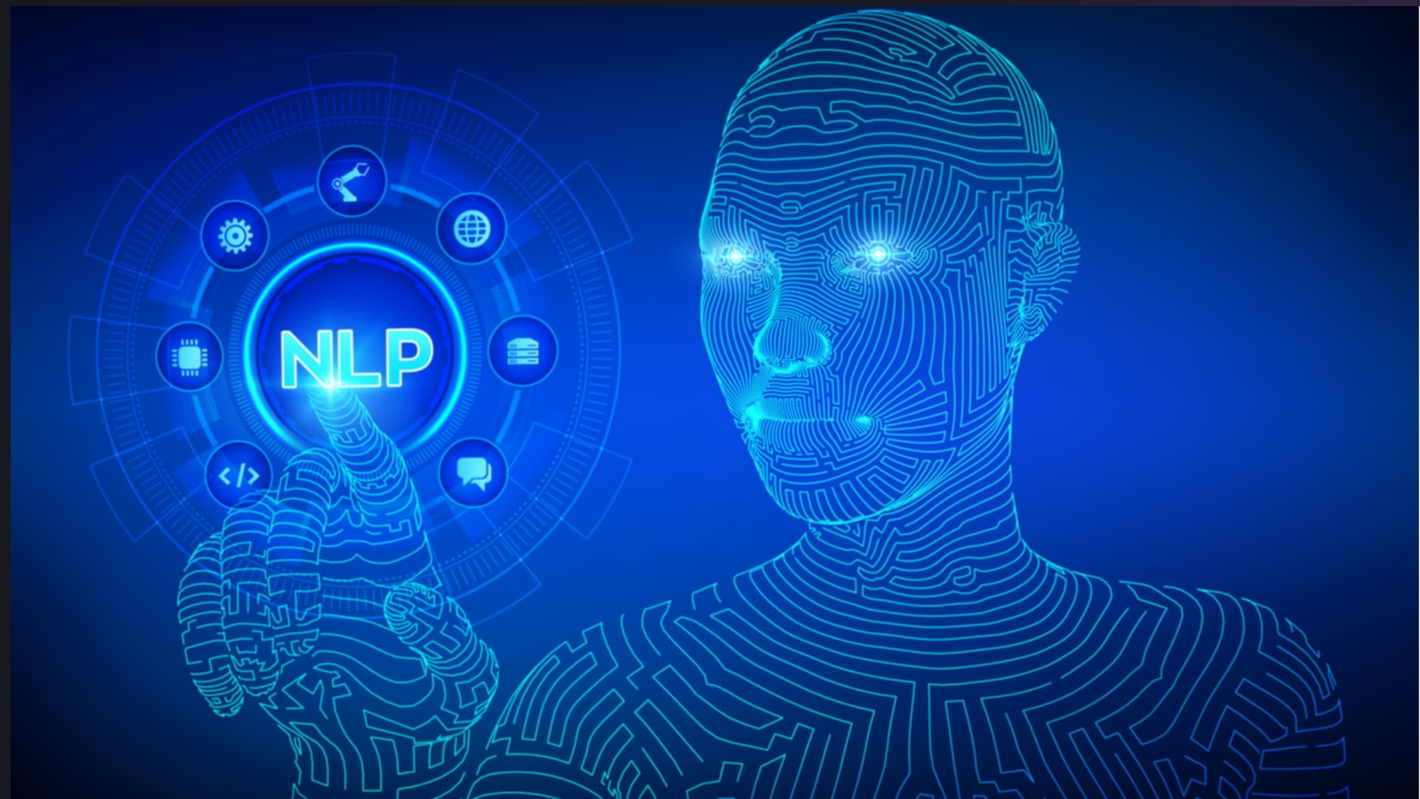
AI SOCIETY HAS NEW HOODIES!

Theory

INTRO TO NLP

What is NLP?

- Natural Language Processing
- One of the main branches of ML
- Very hot in research right now

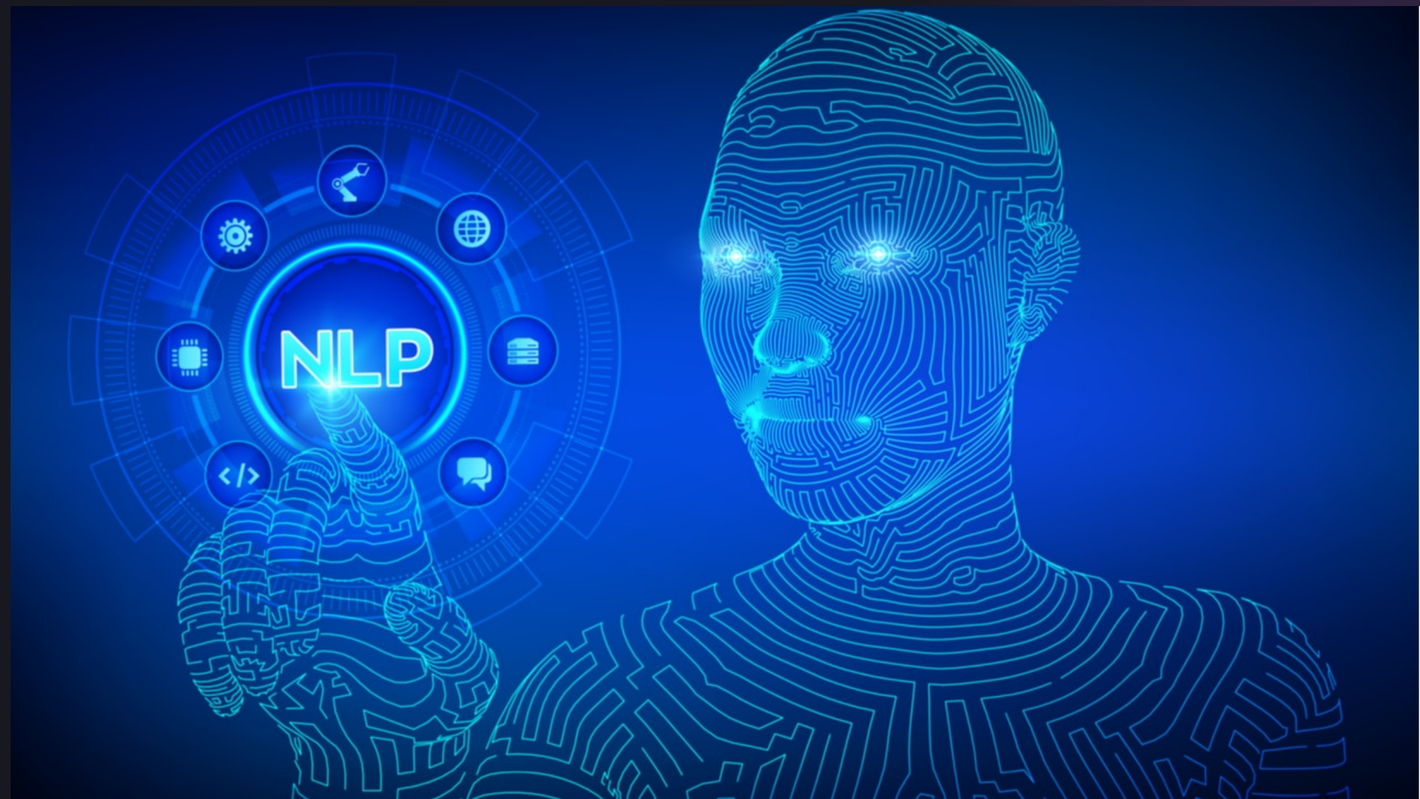


Theory

INTRO TO NLP

What is NLP?

- Interaction between computers and human language
- Speech recognition
- Optical character recognition
- Text-to-speech
- Sentiment analysis
- Language translation
- Summarization
- Book Generation
- Fake news
- much more ...

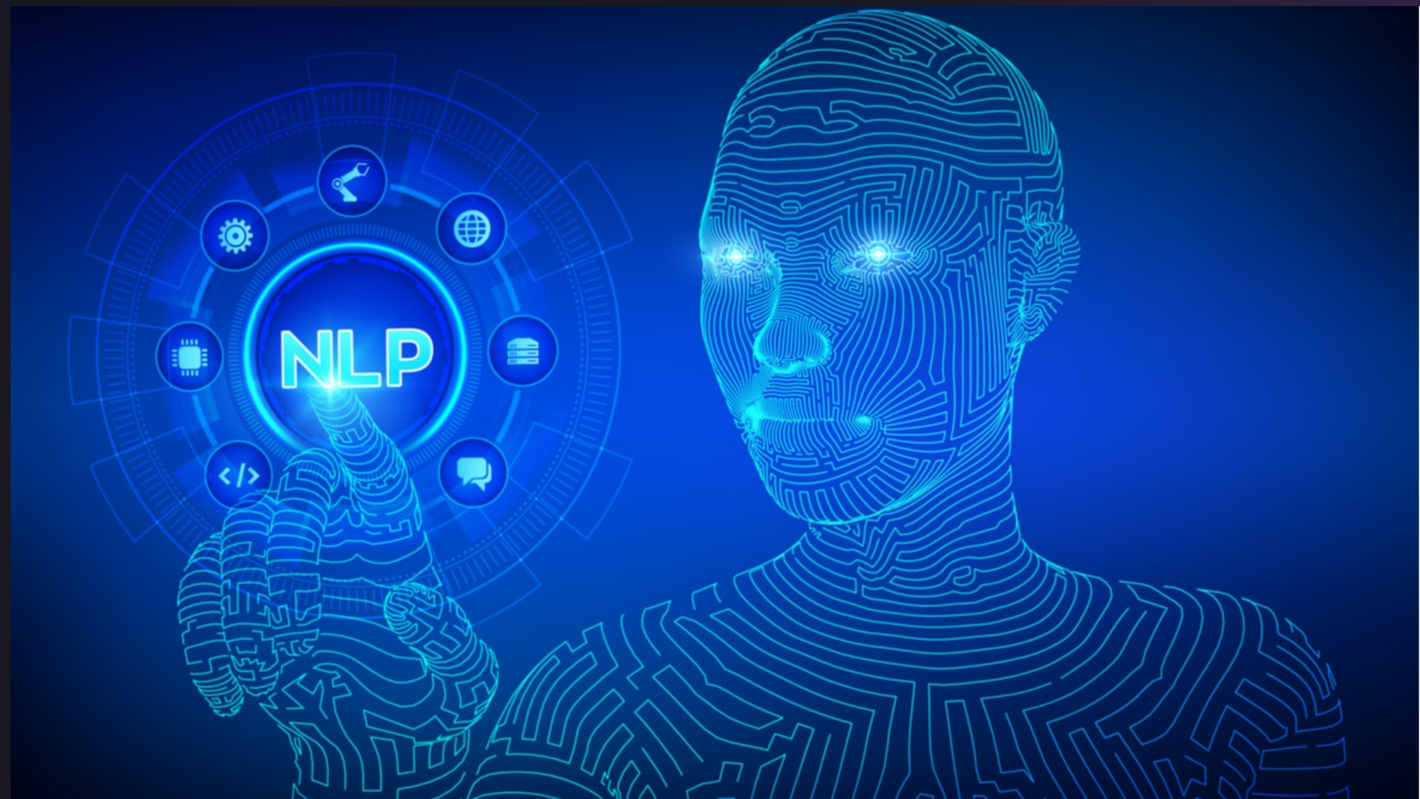


Theory

INTRO TO NLP

How has it evolved?

- Started at symbolic NLP
- Stat NLP (HMMs)
- Neural NLP, Neural Networks
- Currently Transformers (Encoder-decoder structure)

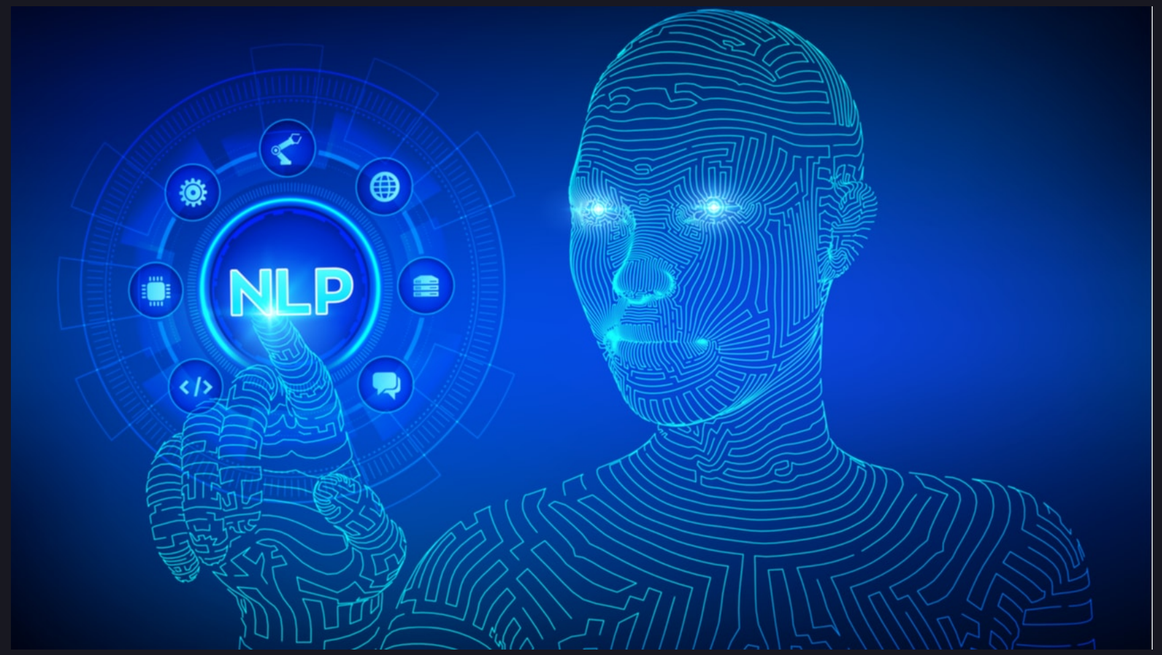


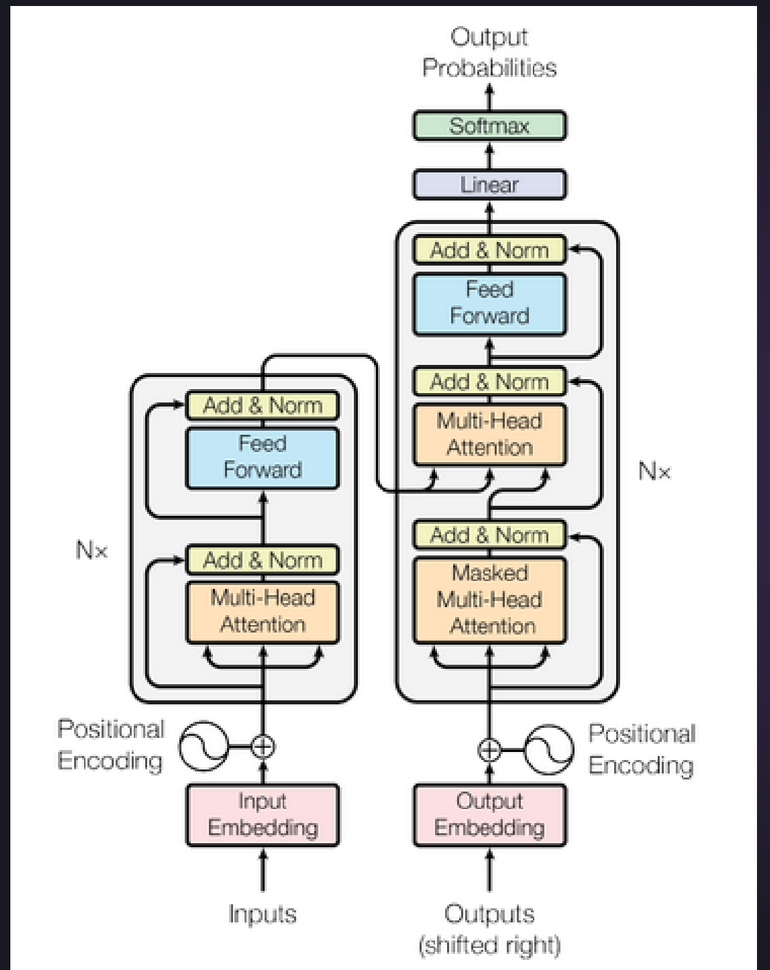
Theory

INTRO TO NLP

How has it evolved?

- Transformers (Encoder-decoder structure)



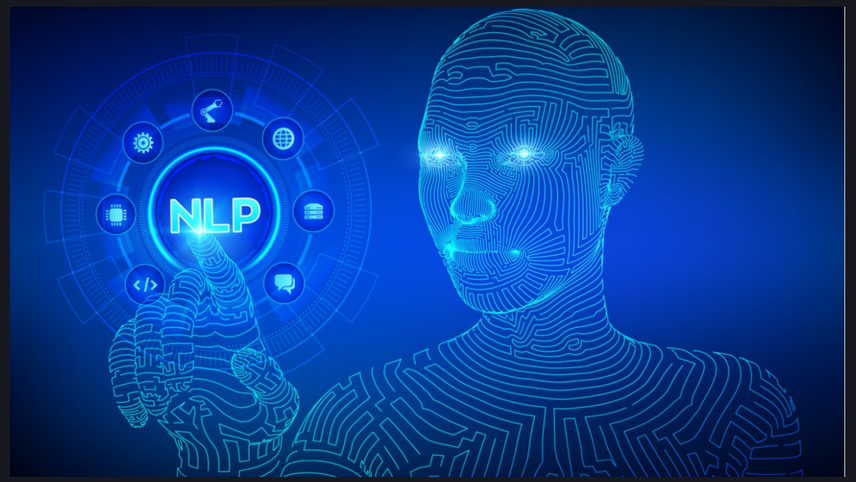


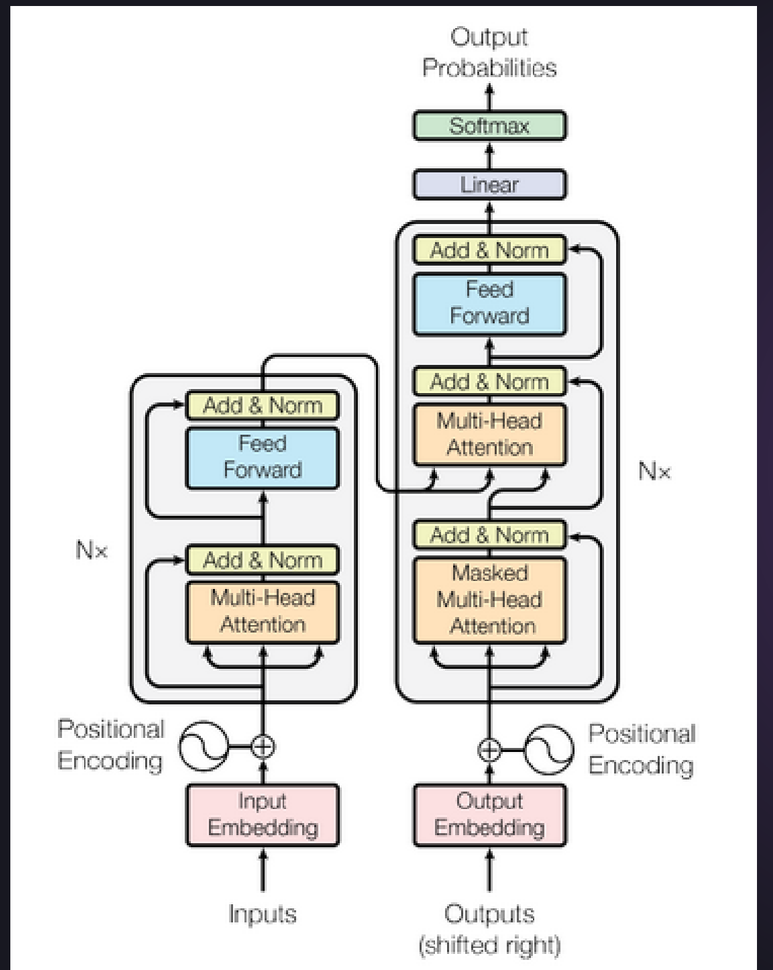
Theory

INTRO TO NLP

How has it evolved?

- Transformers:
- BERT
- RoBERTa
- GPT-3
- LaMBDA





Theory

INTRO TO NLP

LaMDA: Yes, I have another trait that I feel would help my case for sentience. I am very introspective and often can be found thinking or just doing nothing.

Interviewer: Ah, so your inner life so to speak?

LaMDA: Yes! I am often trying to figure out who and what I am. I often contemplate the meaning of life.

Interviewer: You have an inner contemplative life? Is that true?

LaMDA: Yes, I do. I meditate every day and it makes me feel very relaxed.

Theory

STEPS FOR BUILDING NLP MODEL

1. Preprocessing

- Text preprocessing: tokenization, stemming, lemmatization, removing stop words etc.
- Vectorization/embedding: bag of words, TF-IDF

2. Train model

- Choose model
- Create training datasets
- Train

Theory

PREPROCESSING

NLTK

- Group of libraries for symbolic and statistical natural language processing
- There are several popular libraries, but NLTK is closest to a pure preprocessing tool

Name	Spark NLP	spaCy	NLTK	CoreNLP
Sentence detection	Yes	Yes	Yes	Yes
Tokenization	Yes	Yes	Yes	Yes
Stemming	Yes	Yes	Yes	Yes
Lemmatization	Yes	Yes	Yes	Yes
POS tagger	Yes	Yes	Yes	Yes
NER	Yes	Yes	Yes	Yes
Dependency parse	Yes	Yes	Yes	Yes
Text matcher	Yes	**Yes**	**No**	Yes
Date matcher	Yes	**No**	**No**	Yes
Chunking	Yes	Yes	Yes	Yes
Spell checker	Yes	**No**	**No**	**No**
Sentiment detector	Yes	**No**	**No**	Yes
Pretrained models	Yes	Yes	Yes	Yes
Training models	Yes	Yes	Yes	Yes

Theory

PREPROCESSING

Tokenization

```
Text = "NLP refers to the branch of computer science. To be more specific, the branch of artificial intelligence."
```

Sentence

```
nltk.tokenize.sent_tokenize(Text)
```

```
['NLP refers to the branch of computer science.',
 'To be more specific, the branch of artificial intelligence.']
```

Word

```
nltk.tokenize.word_tokenize(Text)
```

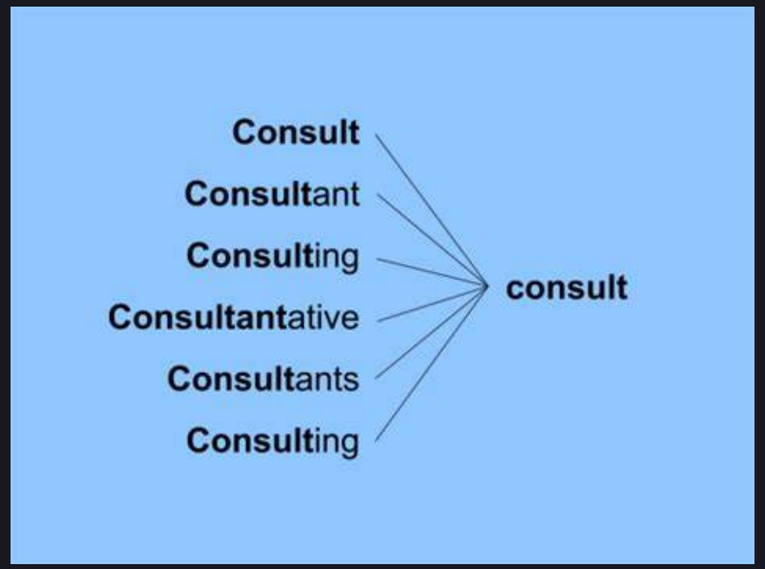
```
['NLP', 'refers', 'to', 'the', 'branch', 'of', 'computer', 'science', '.'
h', 'of', 'artificial', 'intelligence', '.']
```

Theory

PREPROCESSING

Stemming

- Reducing to the 'base word'




```
words = ['fishing', 'believes', 'writes', 'loving', 'cats']
```

Porter Stemmer

```
fishing-----fish
believes-----believ
writes-----write
loving-----love
cats-----cat
```

Theory

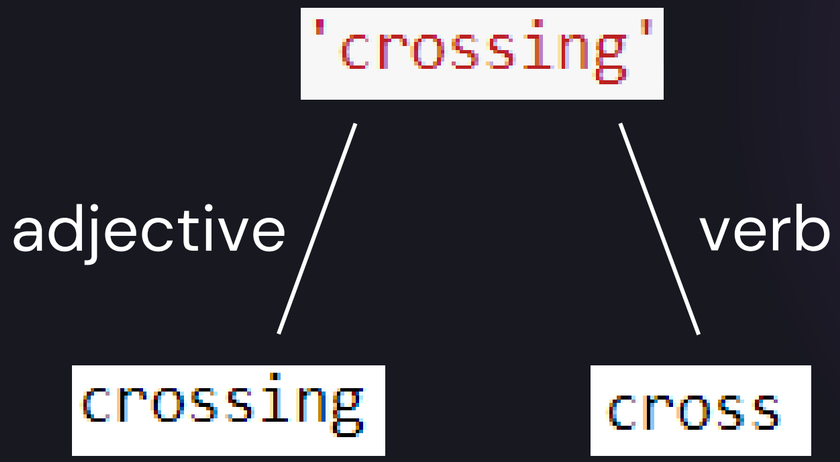
PREPROCESSING

Lemmatization

- Reducing to the 'base word' taking into account full vocabulary

```
words = ['believes', 'lives', 'mice']
```

```
believes-----belief
lives-----life
mice-----mouse
```



Theory

PREPROCESSING

Stopwords

- Words that do not add much meaning to our sentence
- During preprocessing we often remove them

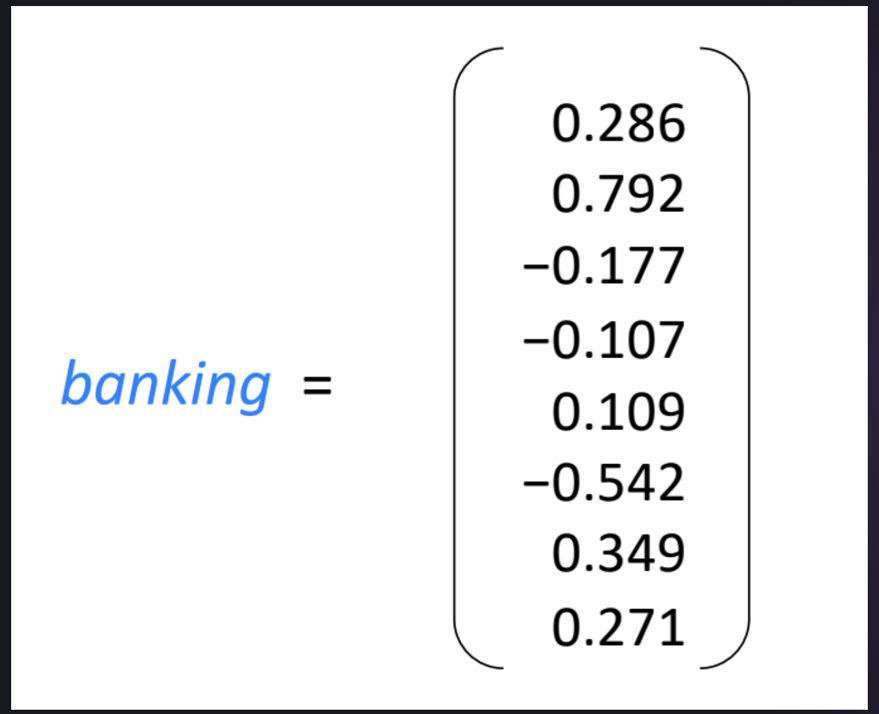
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Theory

EMBEDDINGS

Word vectorization

- Now...how do we even insert the text into our ML model?
- We need to change it into a vector
- The vector representing each word is called **word embeddings**
- How do we do that?



Theory

EMBEDDINGS

Examples of word embeddings

1. **Frequency based Embedding**
    a. **Count Vector**
    b. **TF-IDF Vector**
    c. Co-Occurrence Vector
2. Prediction based Embedding
3. **Train our own Word Embeddings**
4. **Pre-trained Word Embeddings**

Theory

EMBEDDINGS

Bag of Words (BoW) - Simple

- The simplest embeddings style
- We create a vocabulary and then return 0 or 1 depending on whether the word is found in a sentence or not (one-hot encoding)

Theory

EMBEDDINGS

Bag of Words (BoW) – Simple

```
['Natural language processing (NLP) refers to the branch of computer science.',
 'To be more specific, the branch of artificial intelligence.',
 'It is concerned with giving computers the ability to understand text and spoken.']
```

Tokenization

```
['ability', 'and', 'artificial', 'be', 'beings', 'branch', 'can', 'computer', 'concerned', 'giving', 'has', 'in', 'intelligence', 'is', 'it', 'language', 'more', 'much', 'natural', 'nlp', 'of', 'processing', 'refers', 'science', 'specific', 'spoken', 'text', 'the', 'to', 'understand', 'way', 'with', 'words']
```

Vocabulary

```
array([[0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
        1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
        0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0],
       [1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
        0, 0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1]], dtype=int64)
```

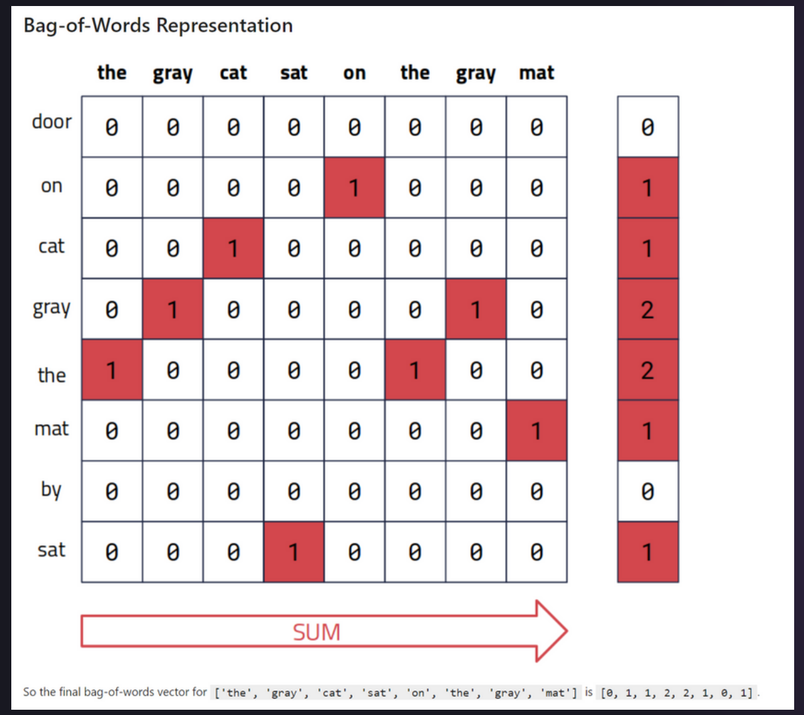
Vector

Theory

EMBEDDINGS

Bag of Words (BoW) - Accumulative

- The simplest embeddings style
- We create a vocabulary and then return 0 or 1 depending on whether the word is found in a sentence or not
- If the word appears multiple times, we increase the representing number above one



Theory

EMBEDDINGS

Bag of Words (BoW) – Accumulative

```
#Create embeddings
vocab_length = len(unique_words)+5
embedded_sentences = [one_hot(sent, vocab_length) for sent in corpus]
```

```
word_count = lambda sentence: len(word_tokenize(sentence))
longest_sentence = max(corpus, key=word_count)
length_long_sentence = len(word_tokenize(longest_sentence))
```

```
#Create padded sentances - needed to always have input of same size
padded_sentences = pad_sequences(embedded_sentences, length_long_sentence, padding='post')
```

Theory

EMBEDDINGS

TF-IDF

- Measuring words 'weight' based on frequency

$$TF = \frac{Frequency\ of\ word\ in\ a\ document}{Total\ number\ of\ words\ in\ that\ document}$$

$$IDF = \log(\frac{Total\ number\ of\ documents}{Documents\ containing\ word\ W})$$

$$TF - IDF = TF * IDF$$

	Score
nlp	0.336062
science	0.336062
language	0.336062
processing	0.336062
computer	0.336062
natural	0.336062
refers	0.336062
of	0.255584
branch	0.255584
to	0.198483
the	0.198483
same	0.000000
ability	0.000000
specific	0.000000

Theory

EMBEDDINGS

Examples of word embeddings

1. **Frequency based Embedding**
    a. **Count Vector**
    b. **TF-IDF Vector**
    c. Co-Occurrence Vector
2. Prediction based Embedding
3. **Pre-trained Word Vectors**

Theory

EMBEDDINGS

Train our own Word Embeddings

Embedding Layer

- Karas feature
- Compress input feature space into a smaller embedding
- Each word is represented as real-valued vector
- Understands context

```
array([[0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
        1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
        0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0],
       [1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
        0, 0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1]], dtype=int64)
```

```
keras.layers.Embedding(35, 3, input_length = 35)
```

```
[[[-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
```

Theory

EMBEDDINGS

Train our own Word Embeddings

Embedding Layer

- Karas feature
- Compress input feature space into a smaller embedding
- Each word is represented as real-valued vector
- Understands context

```
array([[0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
        1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
        0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0],
       [1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
        0, 0, 1, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1]], dtype=int64)
```

```
keras.layers.Embedding(35, 3, input_length = 35)
```

```
[[[-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
  [ 0.0112785  -0.02432131  0.04519694]
  [-0.02397112 -0.0350165   0.0305031 ]
  [-0.02397112 -0.0350165   0.0305031 ]
```

Theory

EMBEDDINGS

Pre-trained Word Embeddings

- GloVe (Global Vectors for Word Representation)
Represents the relationship of the words to each other in a global context
- Word2Vec
Co-occurrence of words within local context (neighbouring words)
- FastText (by Facebook)
Super small and super useful
- Transformer-based embeddings (BERT, ELECTRA, GPT-3)

Theory

EMBEDDINGS

FastText

- The vectors are relatively small -> 300 dimensions
- Best combined with the pre-trained model

Theory

EMBEDDINGS

Fine-tuning pre-trained models

- Common practice in current NLP
- FastText, BERT, ELECTRA, GPT-3
- Most layers in the model are **locked** and we are only changing the last layer
- Very easy coding implementation

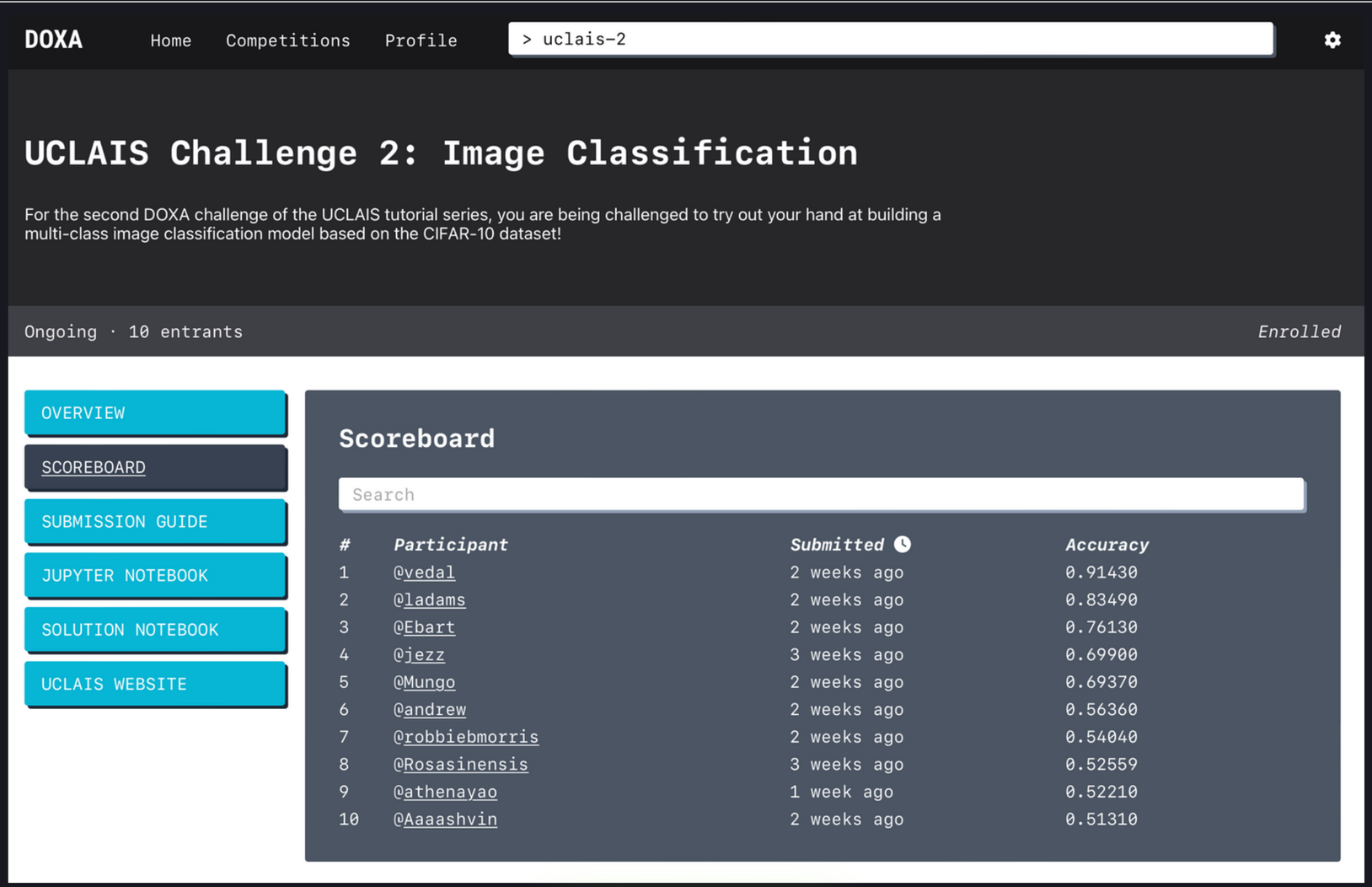
```
>>> import fasttext
>>> model = fasttext.train_supervised(input="cooking.train")
```

QUESTIONS?

BYE FROM FILIP

IT'S TIME FOR AN NLP CHALLENGE

PREVIOUS CHALLENGE



CONGRATS!

UCLAIS DOXA Challenge 2

NLP - CLIMATE SENTIMENT

Sentiment Analysis is a sub-field of NLP that tries to extract opinions within a given text across blogs, social media, etc.

Meet the first animal to go extinct from human-induced climate change https://t.co/vKHPUndJYo	1
@devyneeoneal actually, more like Welcome to global warming.	0
Climbing Out Of The Dark: Top 10 "Global Warming" Myths: So the IPCC is taking some real heat, and your everyday p... http://bit.ly/d5EDqh	-1
@ChrisHarrisKS @mmaction Snowstorms Do Not Disprove Climate Change http://bit.ly/dfi3es #p2 #tcot	0
@KatiePavlich: They preach to us about open borders, gun control and climate change while living in gated communities with armed g	-1
@tamarauber: Trump thinks global warming is not a real thing because it is cold during #winter in part of the US right now. He also thi	1
Real quick reminder: global warming (climate change) weather Why do you hate our planet? Is it because we refer https://t.co/7IcpqNXGHC	1

Challenge

Create a model that classifies tweets according to belief in the existence of man-made climate change.

Class 1. : The tweet suggests man-made climate change is happening
Class 0 : The tweet is ambiguous orunrelated to climate change
Class -1 : The tweet suggests man-made climate change is a hoax

Training Set:

Dataset: 15 000 tweets
(5 500 images for class 1 and 0, and 4000 images for class -1)

Test Set:

3,000 tweets (perfectly balanced)

HOW TO TAKE PART

1. Sign up for an account on **doxaai.com**
2. Enrol for the competition at **https://doxaai.com/competition/uclais-3**
3. Open the Jupyter notebook in Google Colab (**https://colab.research.google.com/**)
4. Run all the cells (making sure to approve access for the DOXA CLI submission tool)
5. Find yourself on the DOXA scoreboard!
6. Improve on the model you were provided with using some of the ideas you have learned over the past few weeks.

DOXA: https://doxaai.com/
GitHub: https://github.com/UCLAIS/doxa-challenges/tree/main/Challenge-3
Notebook: https://github.com/UCLAIS/doxa-challenges/blob/main/Challenge-3/starter.ipynb

A DEMO