Character-based POS tagging

- Presented by Christopher “Mack” McGowen

Problem:

Part-of-speech (POS) tagging for low-inflection languages like English is a mostly-solved problem.

Madlibs!

Example: Madlibs!

1. The _____ DSIR students _____ the instructors.

2. Jonathan is the _____ student in the cohort.

Example: Madlibs!

1. The *ADJ* DSIR students *VERB* the instructors.

2. Jonathan is the *ADJ* student in the cohort.

Example: Madlibs!

1. The grateful DSIR students thanked the instructors.

2. Jonathan is the lamest student in the cohort.

However

Problem:

For highly-inflected languages like ancient Greek, it is more difficult.

Example: Madlibs!

1. The DSIR students _____ the instructors _____.

2. Not provided here because it’s a universal statement.

Example: Madlibs!

1. The DSIR students ____/adj the instructors ____/verb.

2. Not provided here because it's a universal statement.

Goal:

create a model which takes ancient Greek input, and outputs a Part-of-speech (POS) tag(s).

Disclosure:

There are plenty of ancient Greek POS taggers out there, they are all look-up tables with 90%+ accuracy.

Why?

Why?:

POS taggers already exist, but they have 2 major limitations:

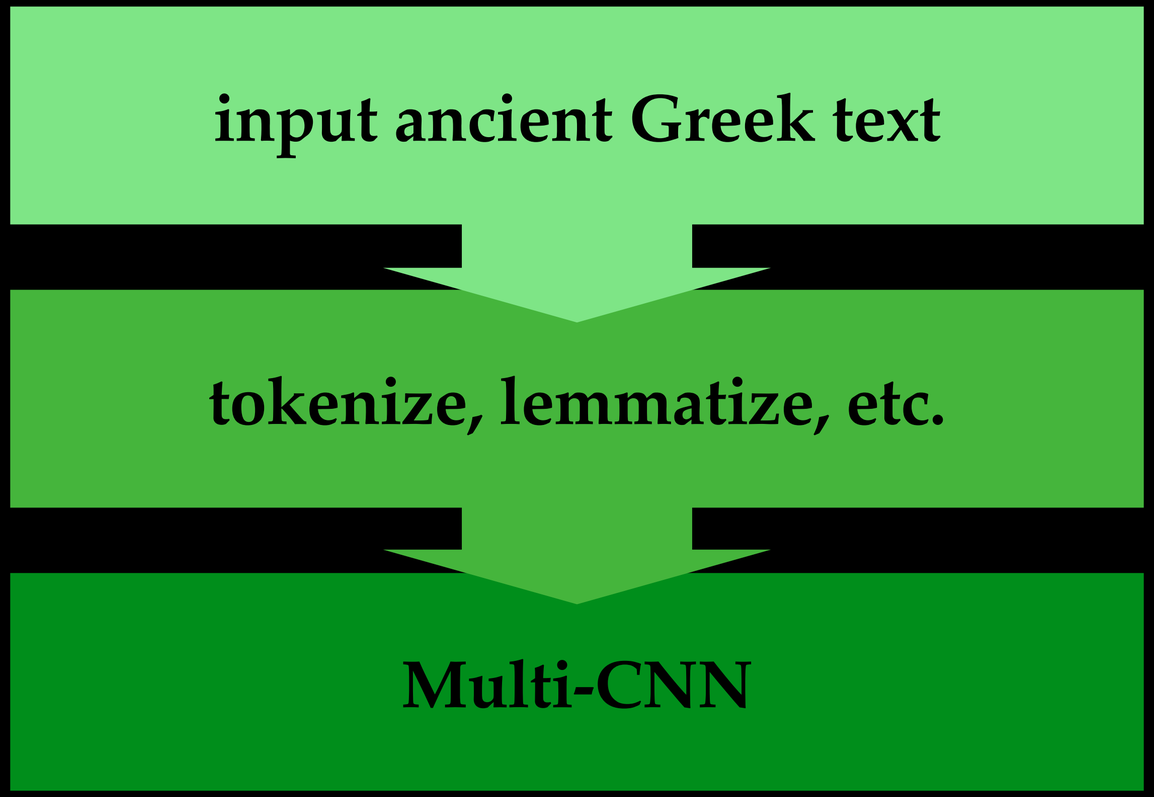
1. They only tag words from their databases.
2. They need a lot of hand-crafted data to work.

Process:

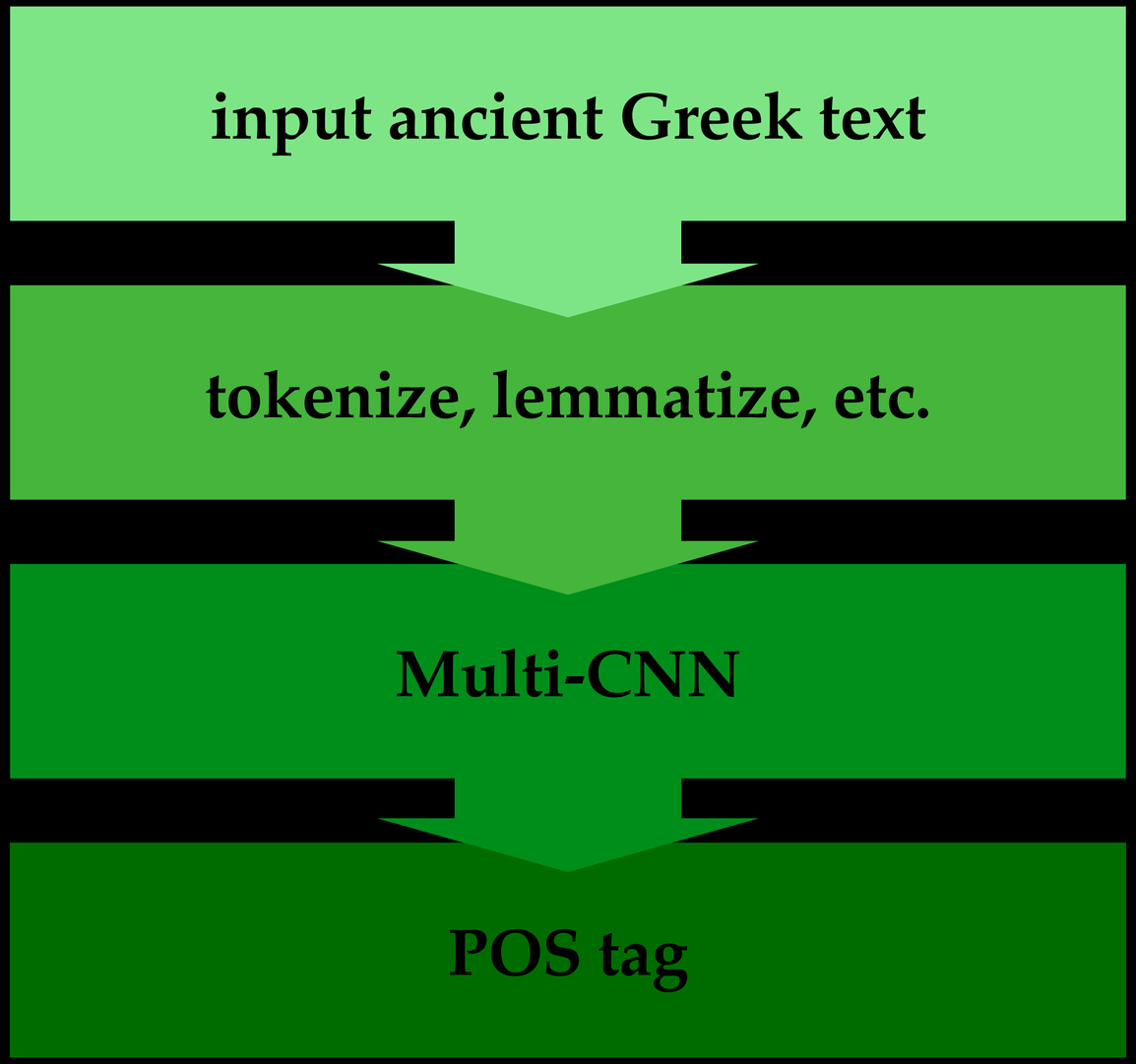
input ancient Greek text

Process:

Process:



Process:


input ancient Greek text
tokenize, lemmatize, etc.
Multi-CNN
POS tag

Process Example:

input ancient Greek text

διπλοῦν ὁρῶσιν οἱ μαθόντες γράμματα

Process Example:

input ancient Greek text

tokenize, lemmatize, etc.

διπλοῦν ὁρῶσιν οἱ μαθόντες γράμματα

Represent each word as a matrix

Process Example:

input ancient Greek text

tokenize, lemmatize, etc.

Multi-CNN

διπλοῦν ὁρῶσιν οἱ μαθόντες γράμματα

Represent each word as a matrix

Use filters to find features (suffix, prefix etc.)

Process Example:

input ancient Greek text

tokenize, lemmatize, etc.

Multi-CNN

POS tag

διπλοῦν ὁρῶσιν οἱ μαθόντες γράμματα

Represent each word as a matrix

Use filters to find features (suffix, prefix etc.)

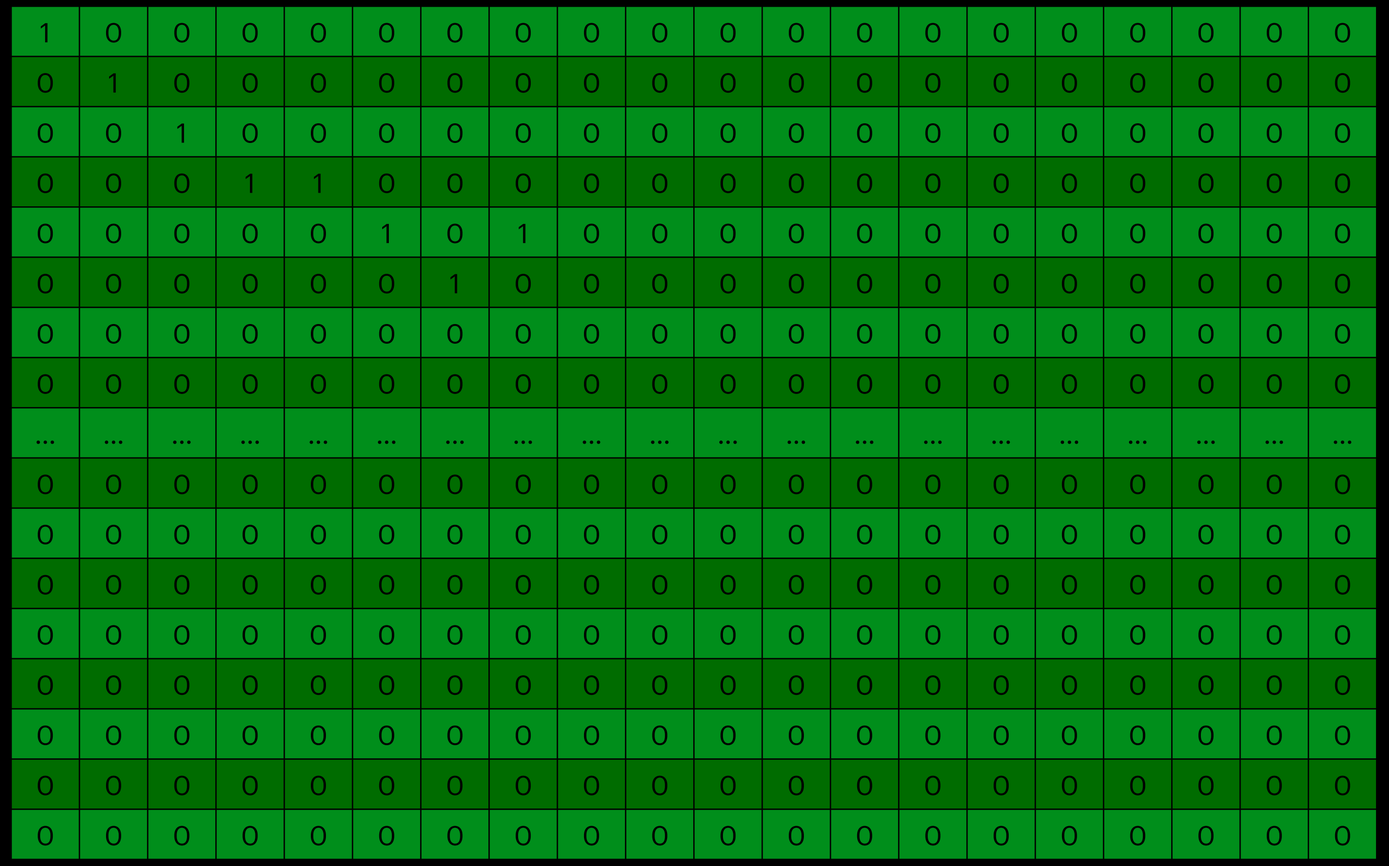
Noun, Verb, etc.

CNN:

20 columns (max word length)

γ ϱ ά μ μ α τ α padding

139 rows
(total possible
letters)

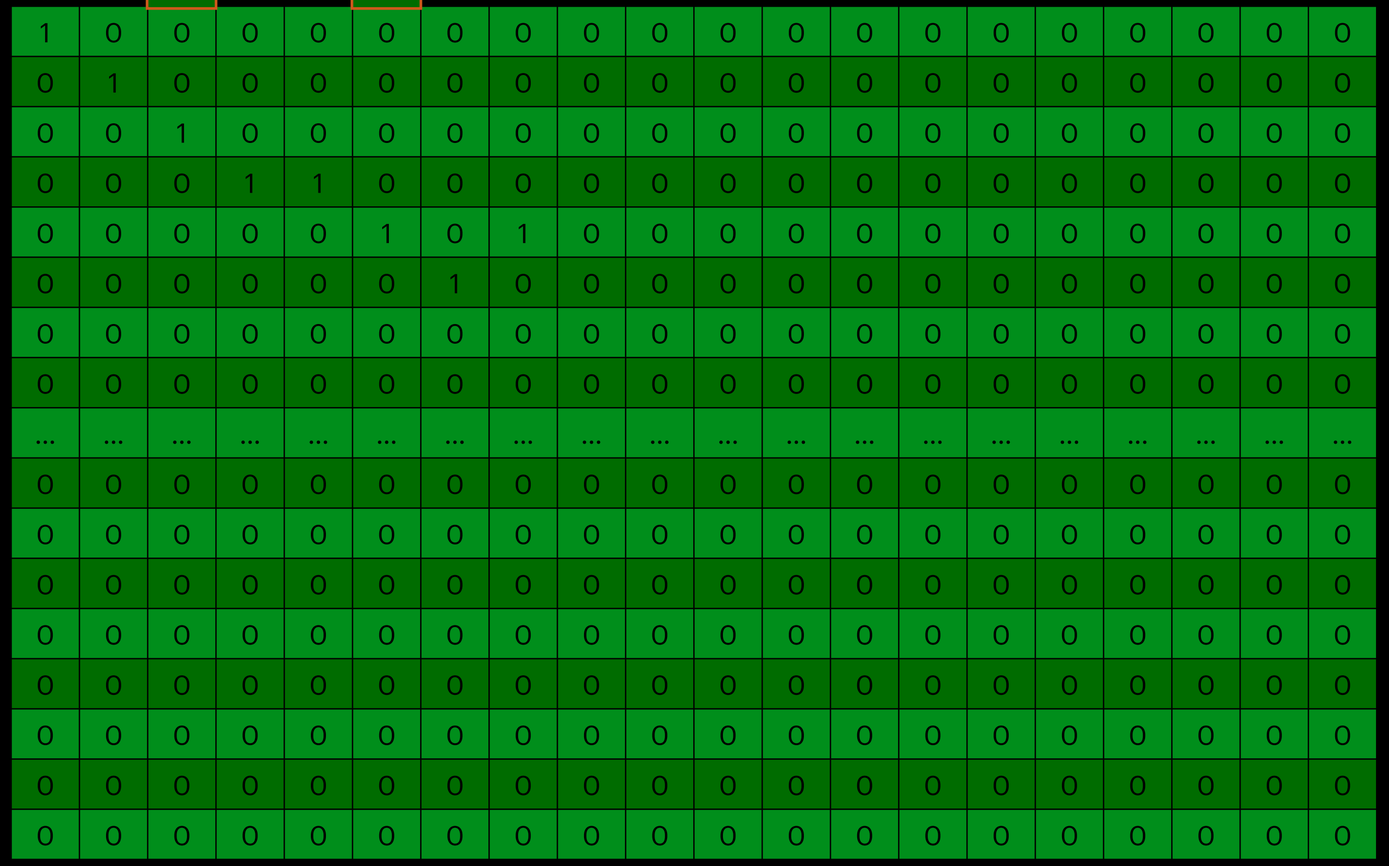


CNN:

20 columns (max word length)

γ ϱ ά μ μ α τ α padding

139 rows
(total possible
letters)



CNN:

20 columns (max word length)

139 rows (total possible letters)

γ	ρ	ά	μ	μ	α	τ	α						padding						
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kernels run = (2,3,4)

CNN:

20 columns (max word length)

139 rows (total possible letters)

γ	ϱ	ά	μ	μ	α	τ	α	padding											
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

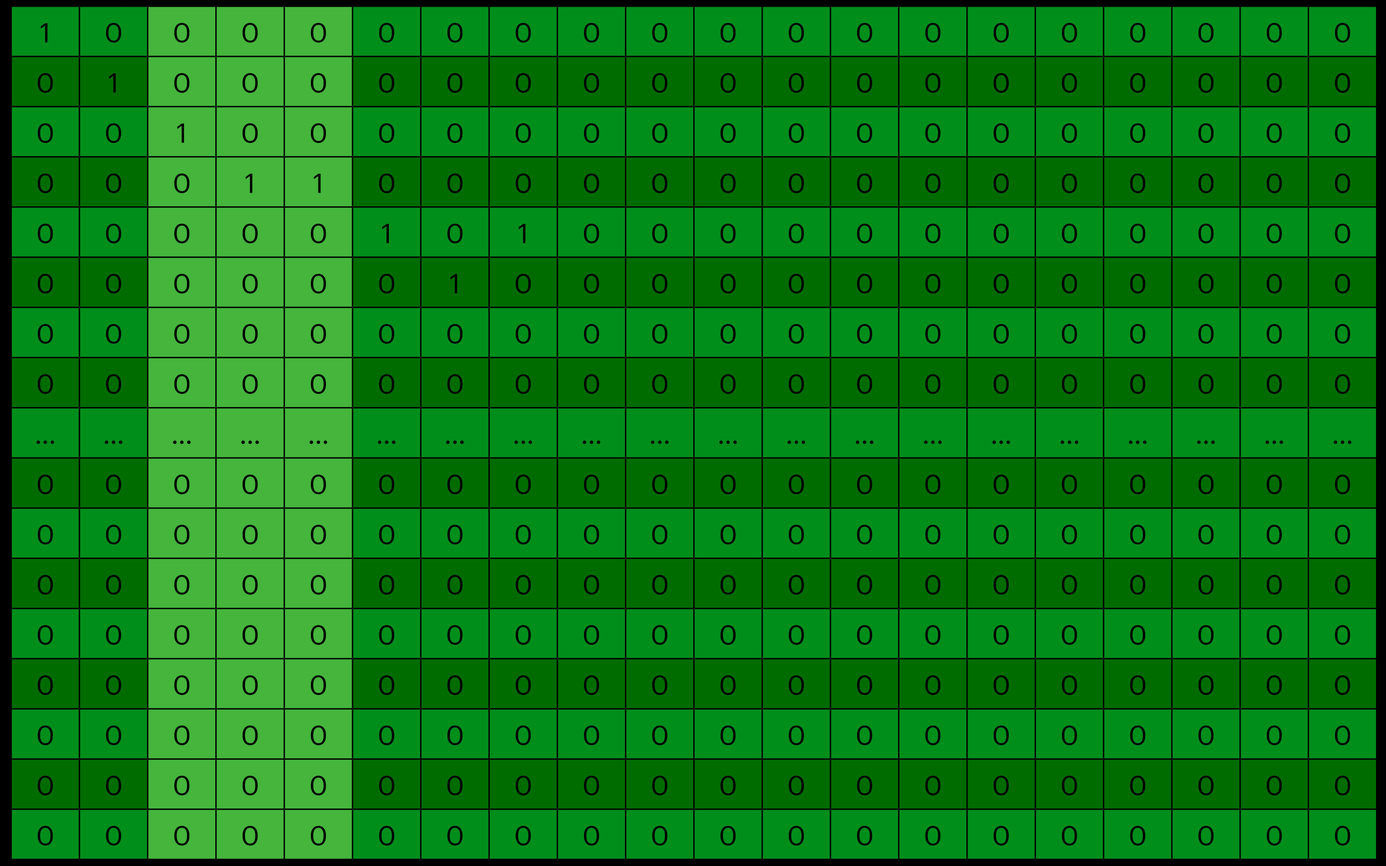
Kernels run = (2,3,4)

CNN:

20 columns (max word length)

γ ϱ ά μ μ α τ α padding

139 rows
(total possible
letters)

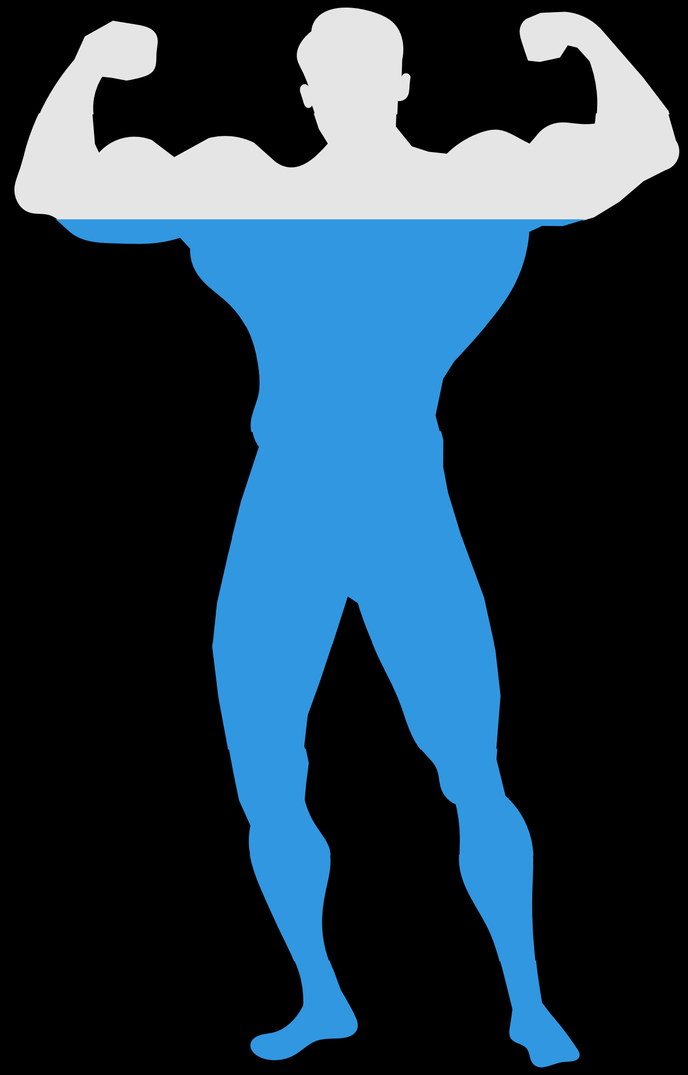


Kernels run = (2,3,4)

Metrics:

85%

Accurate!



Metrics:

POS tag	Precision	F1-score	Count
ADJ	0.80	0.73	7351
ADP	0.68	0.56	224
ADV	0.81	0.64	1083
CCONJ	0.48	0.46	53
DET	0.52	0.35	50
INTJ	0.61	0.46	30
NOUN	0.78	0.82	11579
NUM	0.75	0.33	42
PART	0.50	0.46	96
PRON	0.61	0.65	594
PUNCT	1.00	0.67	10
SCONJ	0.67	0.45	87
VERB	0.93	0.94	17326
X	0.55	0.34	24
Accuracy		85%	38549

Metrics:

POS tag	Precision	F1-score	Count
ADJ	0.80	0.73	7351
ADP	0.68	0.56	224
ADV	0.81	0.64	1083
CCONJ	0.48	0.46	53
DET	0.52	0.35	50
INTJ	0.61	0.46	30
NOUN	0.78	0.82	11579
NUM	0.75	0.33	42
PART	0.50	0.46	96
PRON	0.61	0.65	594
PUNCT	1.00	0.67	10
SCONJ	0.67	0.45	87
VERB	0.93	0.94	17326
X	0.55	0.34	24
Accuracy		85%	38549

Metrics:

POS tag	Precision	F1-score	Count
ADJ	0.80	0.73	7351
ADP	0.68	0.56	224
ADV	0.81	0.64	1083
CCONJ	0.48	0.46	53
DET	0.52	0.35	50
INTJ	0.61	0.46	30
NOUN	0.78	0.82	11579
NUM	0.75	0.33	42
PART	0.50	0.46	96
PRON	0.61	0.65	594
PUNCT	1.00	0.67	10
SCONJ	0.67	0.45	87
VERB	0.93	0.94	17326
X	0.55	0.34	24
Accuracy		85%	38549

Next steps:

- Get more data to balance the classes.
- Words lack context, also use whole sentences.
- Transfer learning from Word2Vec etc.
- Fix the padding.
- Predict more information.
- Incorporate input from annotators.
- Try it on other languages.
- Make web app pretty.

Webapp:

https://dsir-tagger.herokuapp.com/

διπλοῦν ὁρῶσιν οἱ μαθόντες γράμματα

"Those who know the letters see double (twice as much as those who don’t)."

Questions?