Introduction to Natural Language Processing

a.k.a NLP

What is going to happen today

An introduction

Together, building a spam classifier

Your turn, sentiment analysis

Together again, toward improving our spam classifier

A bit about me

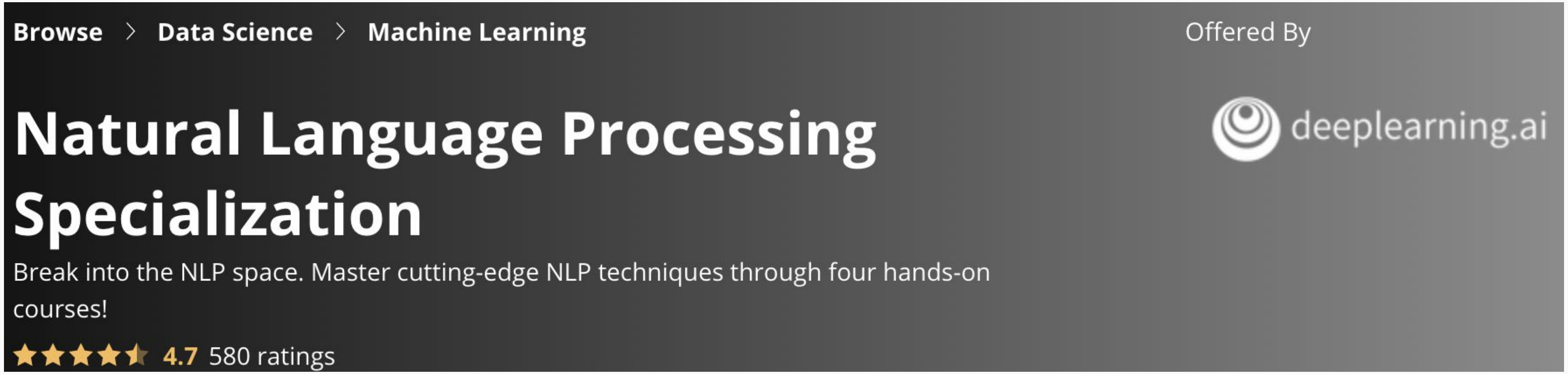
Senior Data Scientist at k_ in Paris

(a FREENOW company)

Curious about NLP and happy to share my recent learnings with you!

What if you fall in love with NLP and want to know more?

deeplearning.ai just launched a new specialization, all about NLP, on Coursera



Browse > Data Science > Machine Learning

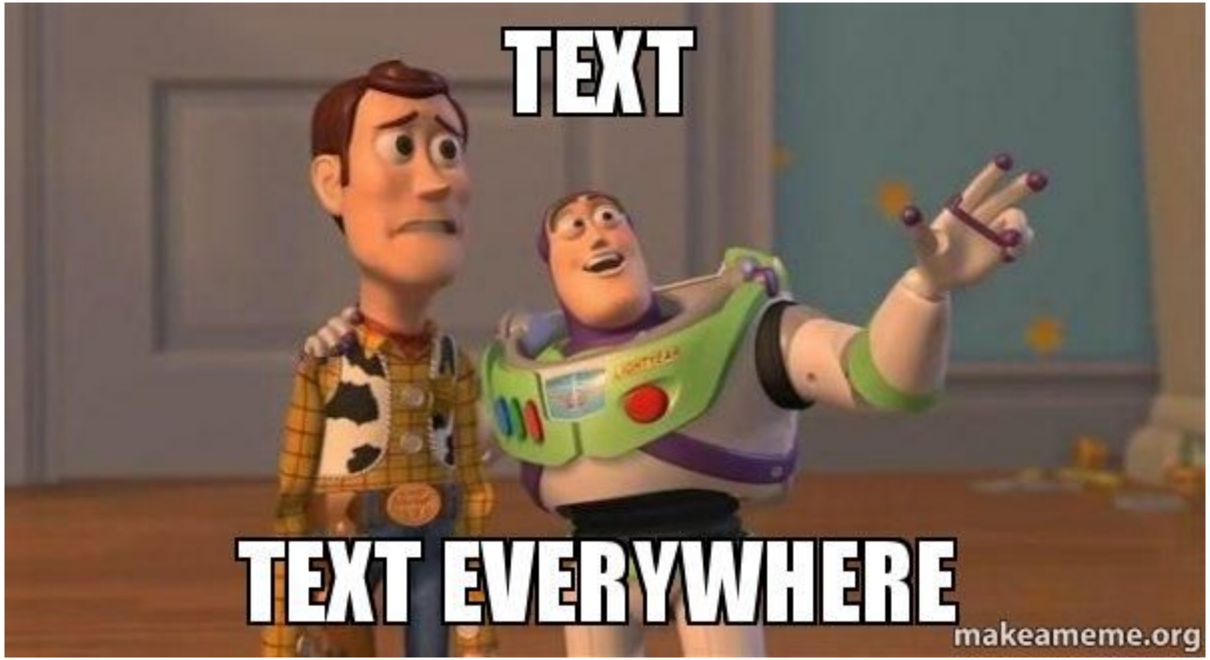
Offered By deeplearning.ai

Natural Language Processing Specialization

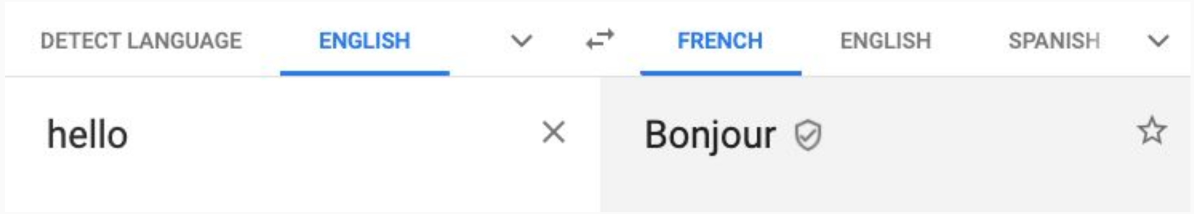
Break into the NLP space. Master cutting-edge NLP techniques through four hands-on courses!

★★★★☆ 4.7 580 ratings

Why care about NLP?


TEXT
TEXT EVERYWHERE
makeameme.org

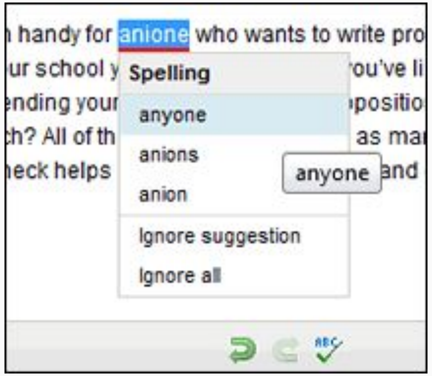
Nice but what for?



Machine translation

*Speech recognition +
Question answering*

Chatbots



Spell checking

How to work with text

How to work with text

Algorithms know well how to work with numbers

so how to with text = how to meaningfully transform text into numbers

Bag-of-words approach

Converting text into numbers

a.k.a Text preprocessing

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

"List listed lists listing listings."

$\Rightarrow$ ['List', 'listed', 'lists', 'listing', 'listings', '.']

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

To "convert a collection of text documents to a matrix of token counts" [1]

[1] Scikit-learn documentation on CountVectorizer

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

An example (from [1])

```
corpus = [
        'This is the first document.',
        'This document is the second document.',
        'And this is the first one.',
        'Is this the first document?',
]
```

```
tokens = ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

Vectorizer output:

```
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

[1] Scikit-learn documentation on CountVectorizer

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

Beware: it gives a lot of weights to frequent (and maybe no so informative) words…

⇒ TF-IDF fixes this

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

TF-IDF: Term Frequency - Inverse Document Frequency

⇒ to measure how important a word is to a document in a corpus

ex: A frequent word in a document that is also frequent in the corpus is less important to a document than a frequent word in a document that is not frequent in the corpus

Text preprocessing

- Tokenization
- CountVectorizer
- **TF-IDF**
- Normalization
- Stemming
- Lemmatization

```
corpus = [
        'This is the first document.',
        'This document is the second document.',
        'And this is the third one.',
        'Is this the first document?',
]
```

```
tokens = ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

Vectorizer output:

```
[[  0, 0.5, 0.6, 0.4,   0,   0, 0.4,   0, 0.4]
 [  0, 0.7,   0, 0.3,   0, 0.5, 0.3,   0, 0.3]
 [0.5,   0,   0, 0.3, 0.5,   0, 0.3, 0.5, 0.3]
 [  0, 0.5, 0.6, 0.4,   0,   0, 0.4,   0, 0.4]]
```

Text preprocessing

Tokenization

<u>**CountVectorizer**</u>

<u>**TF-IDF**</u>

Normalization

Stemming

Lemmatization

CountVectorizer and TF-IDF huge limitation

⇒ corpus dimension *(number of unique tokens)*

To reduce dimension:

- Normalization
- Stemming
- Lemmatization

Text preprocessing

- Tokenization
- CountVectorizer
- TF-IDF
- **Normalization**
- Stemming
- Lemmatization

['List', 'listed', 'lists', 'listing', 'listings', '.']

➡ ['list', 'listed', 'lists', 'listing', 'listings', '.']

Underlying question: do we want to discriminate between “List” and “list”?

Sometimes we do:
⇒ “White House” versus “white house”

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

['list', 'listed', 'lists', 'listing', 'listings', '.']

$\Rightarrow$ ['list', 'list', 'list', 'list', 'list', '.']

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

['list', 'listed', 'lists', 'listing', 'listings', '.']

$\Rightarrow$ ['list', 'listed', 'list', 'listing', 'listing', '.']

Text preprocessing

Tokenization

CountVectorizer

TF-IDF

Normalization

Stemming

Lemmatization

Stemming or Lemmatization?

- It depends :)
- Stemming faster
- Lemmatization more informative

Text classification

Text classification

You can use your favorite classifier with text

Naive Bayes or Logistic Regression usually provide a nice baseline

Performance assessed through AUC Score (Area Under the receiver operating characteristic Curve)

A word about the data we will use

For spam classification

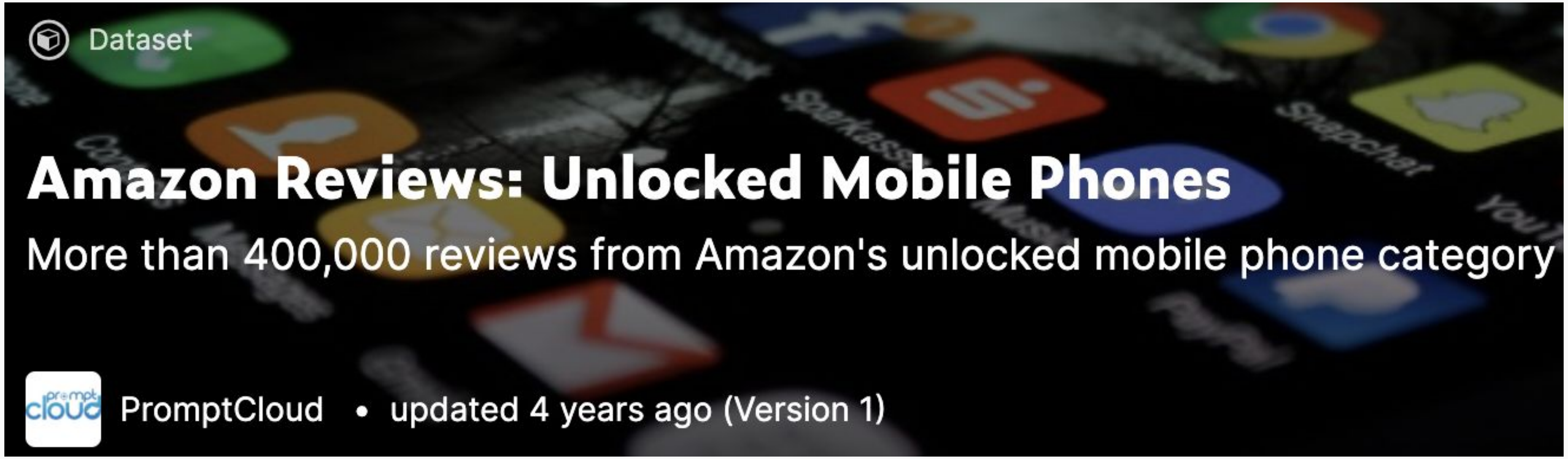
SMS Spam Collection v. 1

“Public set of SMS labeled messages that have been collected for mobile phone spam research. [...] A collection of 5,574 English [...] messages.”

Dedicated web page: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/

Reference paper: Almeida, T.A., Gómez Hidalgo, J.M., Yamakami, A. **Contributions to the Study of SMS Spam Filtering: New Collection and Results.** Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011.

For sentiment analysis



Dataset

Amazon Reviews: Unlocked Mobile Phones

More than 400,000 reviews from Amazon's unlocked mobile phone category

PromptCloud • updated 4 years ago (Version 1)

Available on Kaggle platform

Let’s move on to classifying spam

And now, sentiment analysis!