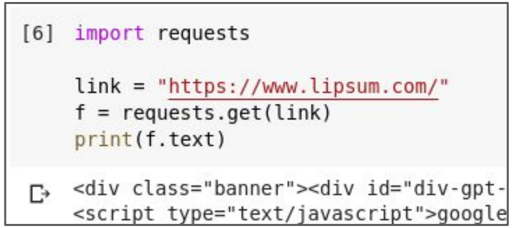
Tools and Knowledge in NLP

Víctor M. Alonso Rorís
victor.roris@dataspartan.com

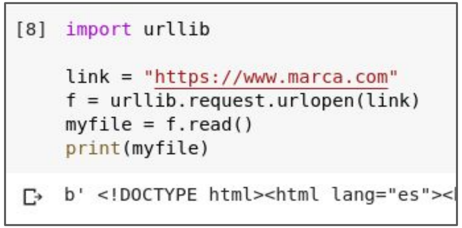
NLP Utils

- Read text from url
  - `urllib`
  - `requests`



```
[6] import requests

    link = "https://www.lipsum.com/"
    f = requests.get(link)
    print(f.text)
```

```
<div class="banner"><div id="div-gpt-
<script type="text/javascript">google
```

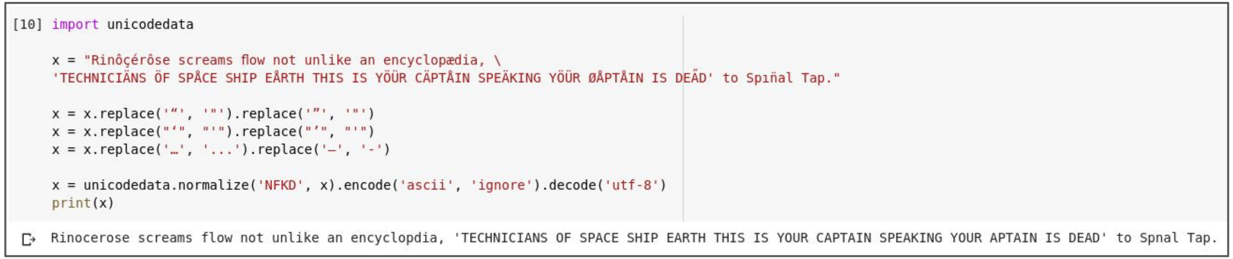


```
[8] import urllib

    link = "https://www.marca.com"
    f = urllib.request.urlopen(link)
    myfile = f.read()
    print(myfile)
```

```
b' <!DOCTYPE html><html lang="es"><
```

- Read from pdfs
  - PDFx [notebook], PyPDF2, PDFMiner, Tika
- HTML Parsers
  - Beautiful Soup: https://www.crummy.com/software/BeautifulSoup
  - Selectolax: https://github.com/rushter/selectolax
- Unicode normalization

Paco Nathan’s proposal ⇒



```
[10] import unicodedata

     x = "Rinôçérôse screams flow not unlike an encyclopædia, \
     'TECHNICIÄNS ÖF SPÅCE SHIP EÅRTH THIS IS YÖÜR CÄPTÅIN SPEÄKING YÖÜR ØÅPTÅIN IS DEĂD' to Spıñal Tap."

     x = x.replace('“', '"').replace('”', '"')
     x = x.replace("‘", "'").replace("’", "'")
     x = x.replace('…', '...').replace('–', '-')

     x = unicodedata.normalize('NFKD', x).encode('ascii', 'ignore').decode('utf-8')
     print(x)
```

```
Rinocerose screams flow not unlike an encyclopdia, 'TECHNICIANS OF SPACE SHIP EARTH THIS IS YOUR CAPTAIN SPEAKING YOUR APTAIN IS DEAD' to Spnal Tap.
```

SpaCy

https://spacy.io/

Is an open-source software library for advanced natural language processing, written in Python and Cython. The library offers statistical neural network models for English, German, Spanish, Portuguese, French, Italian, Dutch and multi-language NER, as well as tokenization for various other languages. It's become one of the most widely used natural language libraries in Python for industry use cases, and has quite a large community — and with that, much support for commercialization of research advances as this area continues to evolve rapidly.

Paco Nathan’s tutorial: https://github.com/DerwenAI/spaCy_tuTorial

SpaCy official course: https://spacy.io/usage/spacy-101

O’Really - Natural Language Processing with Python: https://learning.oreilly.com/videos/natur...

Extensions: https://spacy.io/universe#extensions

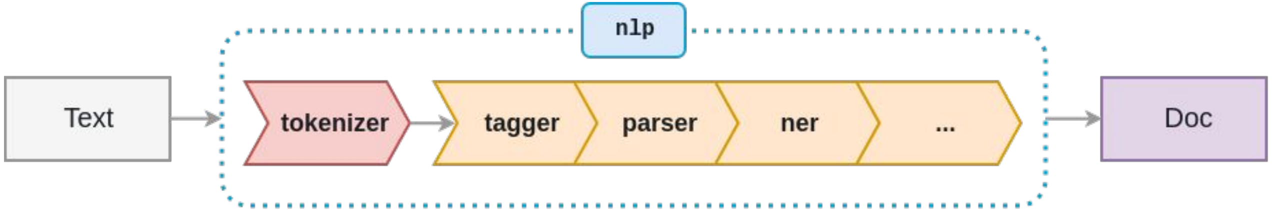
POS (Part of Speech) Tags: https://universaldependencies.org/u/pos/

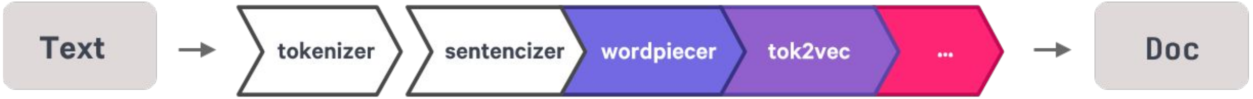
SpaCy - custom pipelines and attributes

https://explosion.ai/blog/spacy-v2-pipelines-extensions
https://github.com/victor-roris/NLPlearning/blob/master/SpaCy/Custom_pipelines_and_extensions_for_spaCy.ipynb

SpaCy 2.0 includes functionalities to extend SpaCy and create your own plugins in a very easy way.

Example: PyTextRank, Spacy-Wordnet, BlackStone, Spacy-HunSpell, ...

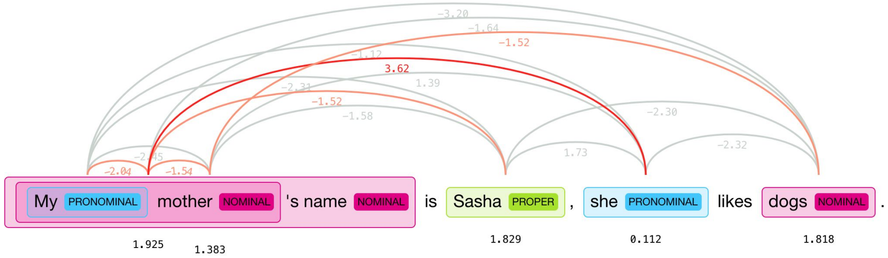




NeuralCoref

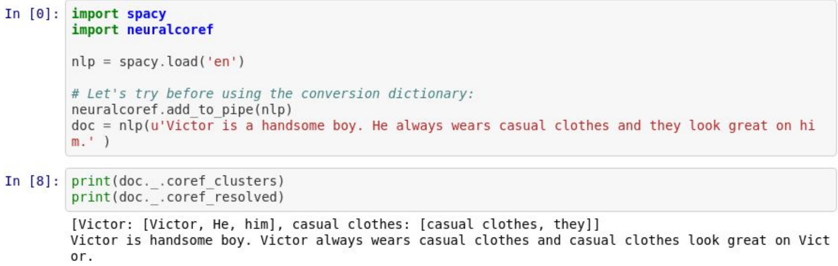
https://medium.com/huggingface/state-of-the-art-neural-coreference-resolution-for-chatbots-3302365dcf30
https://medium.com/huggingface/how-to-train-a-neural-coreference-model-neuralcoref-2-7bb30c1abdfe
https://github.com/huggingface/neuralcoref

coreference: occurs when two or more expressions in a text refer
to the same person or thing.



NeuralCoref is a Hugging Face (explained latter) pipeline extension for spaCy 2.1+ which annotates and resolves coreference clusters using a neural network (see previous slide *custom spaCy pipeline*). It comes with a pre-trained statistical model for **English only**.

Tutorial Notebook:
https://github.com/victor-roris/NLPlearning/blob/master/coreference/NeuralCoref_Coreference_Resolution_use.ipynb



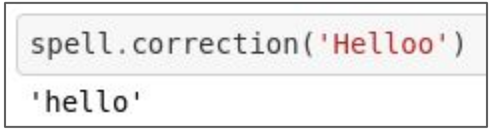
```
In [0]: import spacy
        import neuralcoref

        nlp = spacy.load('en')

        # Let's try before using the conversion dictionary:
        neuralcoref.add_to_pipe(nlp)
        doc = nlp(u'Victor is a handsome boy. He always wears casual clothes and they look great on hi
        m.' )

In [8]: print(doc._.coref_clusters)
        print(doc._.coref_resolved)

        [Victor: [Victor, He, him], casual clothes: [casual clothes, they]]
        Victor is handsome boy. Victor always wears casual clothes and casual clothes look great on Vict
        or.
```

Spell Check

Checking of spelling to text processing or analysis

- **SymSpell:** https://github.com/wolfgarbe/SymSpell [notebook]
- **PyHunSpell:** https://github.com/blatinier/pyhunspell [notebook]
  - **spacy_hunSpell:** pipeline extension for spacy https://github.com/tokestermw/spacy_hunspell [notebook]
- **PySpellChecker:** https://github.com/barrust/pyspellchecker [notebook]

How to write your own spelling corrector: http://norvig.com/spell-correct.html



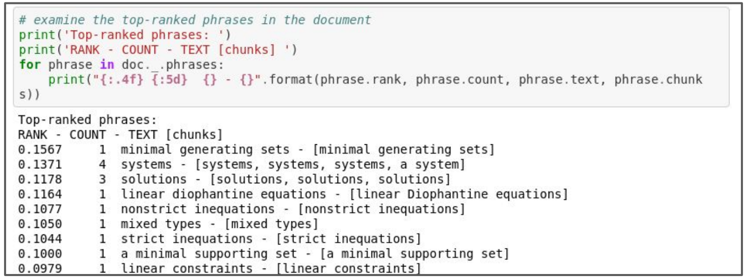
```
spell.correction('Helloo')
'hello'
```

PyTextRank (I)

https://github.com/DerwenAI/pytextrank
https://github.com/victor-roris/NLPlearning/blob/master/summarization-keywords/PyTextRank.ipynb
https://github.com/victor-roris/NLPlearning/blob/master/summarization-keywords/PyTextRank%20-%20Example%20of%20use.ipynb

PyTextRank is a Paco Nathan’s Python implementation of TextRank as a spaCy extension, used to:

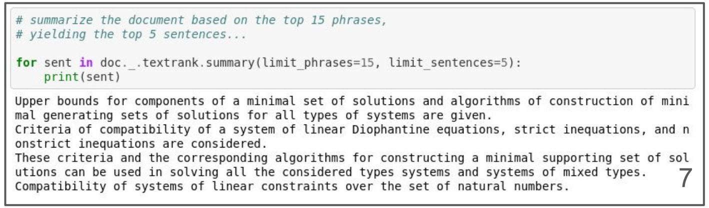
- extract the top-ranked phrases from text documents
- infer links from unstructured text into structured data
- run extractive summarization of text documents



```
# examine the top-ranked phrases in the document
print('Top-ranked phrases: ')
print('RANK - COUNT - TEXT [chunks] ')
for phrase in doc._.phrases:
    print("{:.4f} {:5d}  {} - {}".format(phrase.rank, phrase.count, phrase.text, phrase.chunk
s))
```

```
Top-ranked phrases:
RANK - COUNT - TEXT [chunks]
0.1567      1  minimal generating sets - [minimal generating sets]
0.1371      4  systems - [systems, systems, systems, a system]
0.1178      3  solutions - [solutions, solutions, solutions]
0.1164      1  linear diophantine equations - [linear Diophantine equations]
0.1077      1  nonstrict inequations - [nonstrict inequations]
0.1050      1  mixed types - [mixed types]
0.1044      1  strict inequations - [strict inequations]
0.1000      1  a minimal supporting set - [a minimal supporting set]
0.0979      1  linear constraints - [linear constraints]
```

The idea is get the noun_chunks (ex., The European Commision) and calculate relations between them. With that, it try to find what words/phrases are more important by its popularity (number of relations). Based on PageRange.
Summarization take the sentences with a high score, based on: sentence with more textrank words/phrases and the importance of each of them.



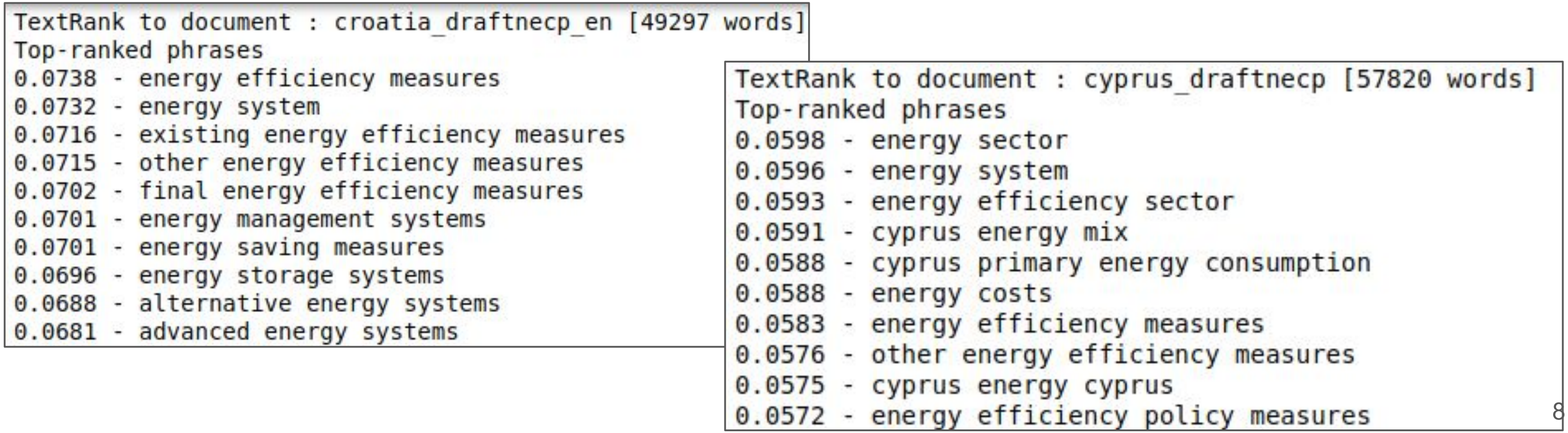
```
# summarize the document based on the top 15 phrases,
# yielding the top 5 sentences...

for sent in doc._.textrank.summary(limit_phrases=15, limit_sentences=5):
    print(sent)
```

```
Upper bounds for components of a minimal set of solutions and algorithms of construction of mini
mal generating sets of solutions for all types of systems are given.
Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and n
onstrict inequations are considered.
These criteria and the corresponding algorithms for constructing a minimal supporting set of sol
utions can be used in solving all the considered types systems and systems of mixed types.
Compatibility of systems of linear constraints over the set of natural numbers.
```

PyTextRank (II)

PyTextRank was applied over the Reganosa documents.

Exists some problems because the raw text extracted from the reganosa documents contains multiple errors and problems (ex. union of sentences, misspelling words, etc.). The result was interesting: the *keywords* seems appropriate (ex., natural gas, projects, name of countries, etc.).



```
TextRank to document : croatia_draftnecp_en [49297 words]
Top-ranked phrases
0.0738 - energy efficiency measures
0.0732 - energy system
0.0716 - existing energy efficiency measures
0.0715 - other energy efficiency measures
0.0702 - final energy efficiency measures
0.0701 - energy management systems
0.0701 - energy saving measures
0.0696 - energy storage systems
0.0688 - alternative energy systems
0.0681 - advanced energy systems
```

```
TextRank to document : cyprus_draftnecp [57820 words]
Top-ranked phrases
0.0598 - energy sector
0.0596 - energy system
0.0593 - energy efficiency sector
0.0591 - cyprus energy mix
0.0588 - cyprus primary energy consumption
0.0588 - energy costs
0.0583 - energy efficiency measures
0.0576 - other energy efficiency measures
0.0575 - cyprus energy cyprus
0.0572 - energy efficiency policy measures
```

Spacy Wordnet

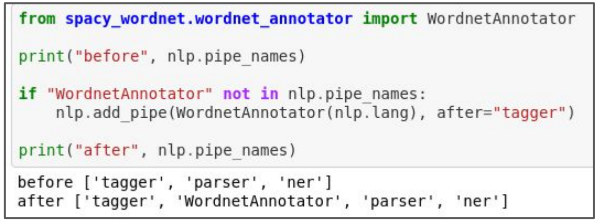
https://github.com/recognai/spacy-wordnet
https://github.com/victor-roris/NLPlearning/blob/master/SpaCy/SpacyWordnet.ipynb

spaCy Wordnet is a simple custom component for using WordNet, MultiWordnet and WordNet domains with spaCy.

The component combines the NLTK wordnet interface with WordNet domains to allow users to:

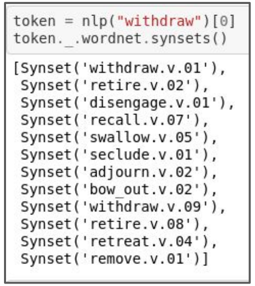
- Get all synsets for a processed token. For example, getting all the synsets (word senses) of the word `bank`.
- Get and filter synsets by domain. For example, getting synonyms of the verb `withdraw` in the financial domain.

This component is integrated in spaCy as a component of a custom pipeline.



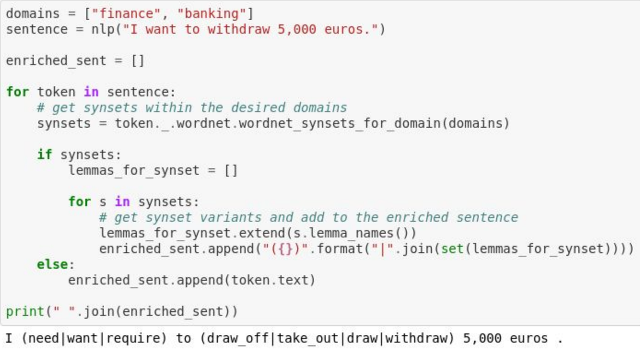
```
from spacy_wordnet.wordnet_annotator import WordnetAnnotator

print("before", nlp.pipe_names)

if "WordnetAnnotator" not in nlp.pipe_names:
    nlp.add_pipe(WordnetAnnotator(nlp.lang), after="tagger")

print("after", nlp.pipe_names)
```

```
before ['tagger', 'parser', 'ner']
after ['tagger', 'WordnetAnnotator', 'parser', 'ner']
```



```
token = nlp("withdraw")[0]
token._.wordnet.synsets()
```

```
[Synset('withdraw.v.01'),
 Synset('retire.v.02'),
 Synset('disengage.v.01'),
 Synset('recall.v.07'),
 Synset('swallow.v.05'),
 Synset('seclude.v.01'),
 Synset('adjourn.v.02'),
 Synset('bow_out.v.02'),
 Synset('withdraw.v.09'),
 Synset('retire.v.08'),
 Synset('retreat.v.04'),
 Synset('remove.v.01')]
```



```
domains = ["finance", "banking"]
sentence = nlp("I want to withdraw 5,000 euros.")

enriched_sent = []

for token in sentence:
    # get synsets within the desired domains
    synsets = token._.wordnet.wordnet_synsets_for_domain(domains)

    if synsets:
        lemmas_for_synset = []

        for s in synsets:
            # get synset variants and add to the enriched sentence
            lemmas_for_synset.extend(s.lemma_names())
            enriched_sent.append("({})".format("|".join(set(lemmas_for_synset))))
    else:
        enriched_sent.append(token.text)

print(" ".join(enriched_sent))
```

```
I (need|want|require) to (draw_off|take_out|draw|withdraw) 5,000 euros .
```

SpaCy - NER training

Named-Entity Recognition component in the SpaCy pipeline can be trained to identify new types of entities. You need labelled examples.

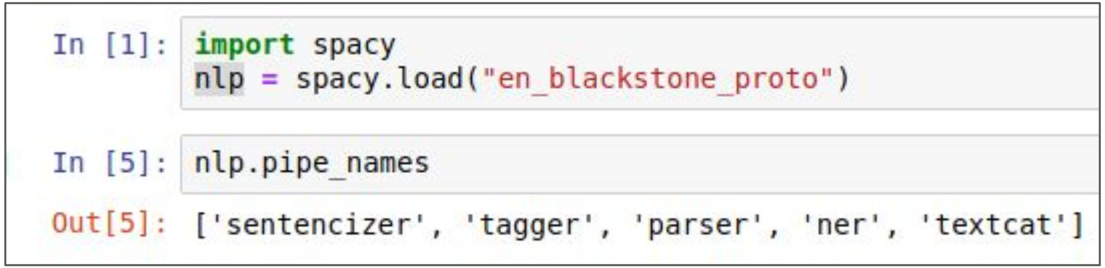
Example NER training:

- *Automatic Summarization of Resumes using NER.*
    - https://dataturks.com/blog/named-entity-recognition-in-resumes.php
    - https://github.com/DataTurks-Engg/Entity-Recognition-In-Resumes-SpaCy/blob/master/train.py
    - https://github.com/victor-roris/NLPlearning/blob/master/NER/Train_NER_spaCy_Model.ipynb
- Train a new entity type
    - https://github.com/explosion/spaCy/blob/master/examples/training/train_new_entity_type.py

Blackstone (I)

https://research.iclr.co.uk/blackstone
https://github.com/ICLRandD/Blackstone

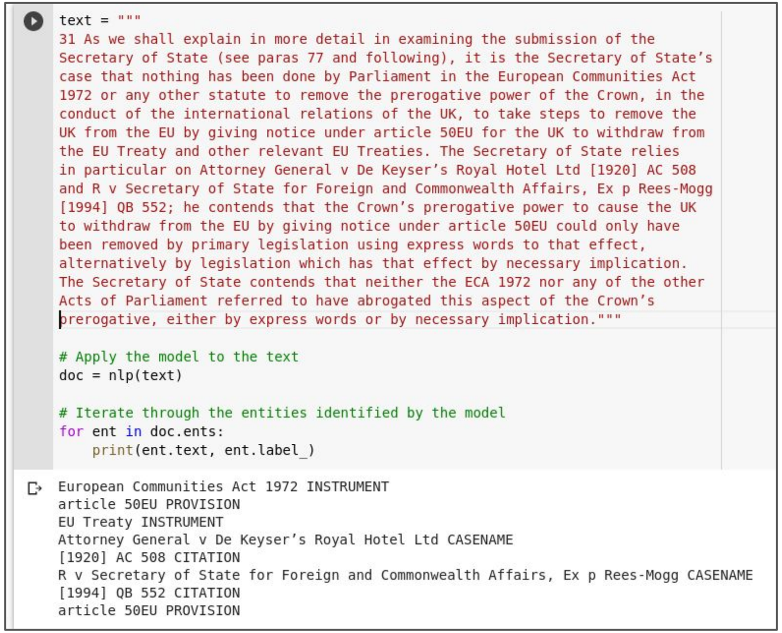
Blackstone is a spaCy model and library for processing long-form, unstructured legal text.
This model includes custom NER and Text Categoriser (and Abbreviation detector).



```
In [1]: import spacy
        nlp = spacy.load("en_blackstone_proto")

In [5]: nlp.pipe_names

Out[5]: ['sentencizer', 'tagger', 'parser', 'ner', 'textcat']
```

Blackstone (II)

- *Named-Entity Recogniser (NER)*
  - CASENAME: Case names
  - CITATION: Citations (unique identifiers for reported and unreported cases)
  - INSTRUMENT: Written legal instruments
  - PROVISION: Unit within a written legal instrument
  - COURT: Court or tribunal
  - JUDGE: References to judges
- *Text Categoriser*: the text categoriser classifies longer spans of text, such as sentences.
  - AXIOM: The text appears to postulate a well-established principle
  - CONCLUSION: The text appears to make a finding, holding, determination or conclusion
  - ISSUE: The text appears to discuss an issue or question
  - LEGAL_TEST: The test appears to discuss a legal test
  - UNCAT: The text does not fall into one of the four categories above



```
text = """
31 As we shall explain in more detail in examining the submission of the
Secretary of State (see paras 77 and following), it is the Secretary of State's
case that nothing has been done by Parliament in the European Communities Act
1972 or any other statute to remove the prerogative power of the Crown, in the
conduct of the international relations of the UK, to take steps to remove the
UK from the EU by giving notice under article 50EU for the UK to withdraw from
the EU Treaty and other relevant EU Treaties. The Secretary of State relies
in particular on Attorney General v De Keyser's Royal Hotel Ltd [1920] AC 508
and R v Secretary of State for Foreign and Commonwealth Affairs, Ex p Rees-Mogg
[1994] QB 552; he contends that the Crown's prerogative power to cause the UK
to withdraw from the EU by giving notice under article 50EU could only have
been removed by primary legislation using express words to that effect,
alternatively by legislation which has that effect by necessary implication.
The Secretary of State contends that neither the ECA 1972 nor any of the other
Acts of Parliament referred to have abrogated this aspect of the Crown's
prerogative, either by express words or by necessary implication."""

# Apply the model to the text
doc = nlp(text)

# Iterate through the entities identified by the model
for ent in doc.ents:
    print(ent.text, ent.label_)
```

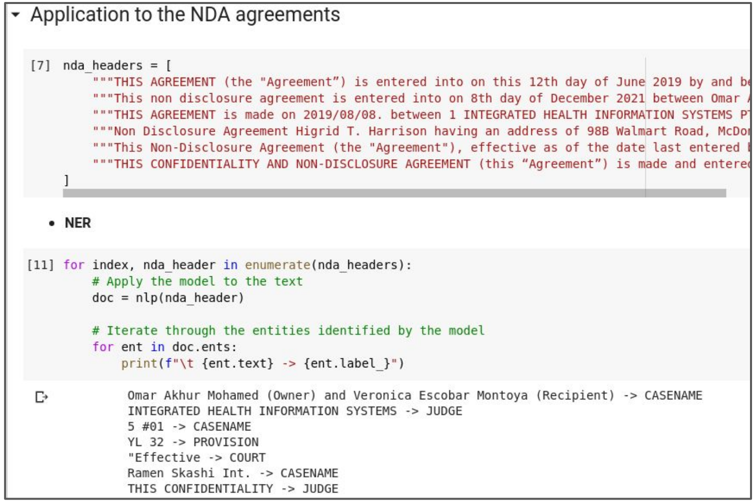
```
European Communities Act 1972 INSTRUMENT
article 50EU PROVISION
EU Treaty INSTRUMENT
Attorney General v De Keyser's Royal Hotel Ltd CASENAME
[1920] AC 508 CITATION
R v Secretary of State for Foreign and Commonwealth Affairs, Ex p Rees-Mogg CASENAME
[1994] QB 552 CITATION
article 50EU PROVISION
```

Blackstone (III)

Blackstone was tested in Agreement texts. No valid results.

https://github.com/victor-roris/NLPlearning/blob/master/spacy_models/Blackstone_Spacy_model_for_legal_texts.ipynb

Conclusion: **Blackstone is a model to court documents and laws**.



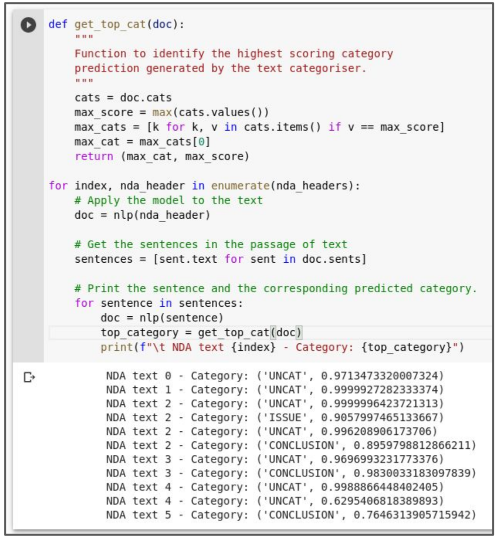
Application to the NDA agreements

```
[7]  nda_headers = [
         """THIS AGREEMENT (the "Agreement") is entered into on this 12th day of June 2019 by and be
         """This non disclosure agreement is entered into on 8th day of December 2021 between Omar A
         """THIS AGREEMENT is made on 2019/08/08. between 1 INTEGRATED HEALTH INFORMATION SYSTEMS PT
         """Non Disclosure Agreement Higrid T. Harrison having an address of 98B Walmart Road, McDor
         """This Non-Disclosure Agreement (the "Agreement"), effective as of the date last entered b
         """THIS CONFIDENTIALITY AND NON-DISCLOSURE AGREEMENT (this "Agreement") is made and entere
     ]
```

- **NER**

```
[11] for index, nda_header in enumerate(nda_headers):
         # Apply the model to the text
         doc = nlp(nda_header)

         # Iterate through the entities identified by the model
         for ent in doc.ents:
             print(f"\t {ent.text} -> {ent.label_}")
```

```
        Omar Akhur Mohamed (Owner) and Veronica Escobar Montoya (Recipient) -> CASENAME
        INTEGRATED HEALTH INFORMATION SYSTEMS -> JUDGE
        5 #01 -> CASENAME
        YL 32 -> PROVISION
        "Effective -> COURT
        Ramen Skashi Int. -> CASENAME
        THIS CONFIDENTIALITY -> JUDGE
```



```
def get_top_cat(doc):
    """
    Function to identify the highest scoring category
    prediction generated by the text categoriser.
    """
    cats = doc.cats
    max_score = max(cats.values())
    max_cats = [k for k, v in cats.items() if v == max_score]
    max_cat = max_cats[0]
    return (max_cat, max_score)

for index, nda_header in enumerate(nda_headers):
    # Apply the model to the text
    doc = nlp(nda_header)

    # Get the sentences in the passage of text
    sentences = [sent.text for sent in doc.sents]

    # Print the sentence and the corresponding predicted category.
    for sentence in sentences:
        doc = nlp(sentence)
        top_category = get_top_cat(doc)
        print(f"\t NDA text {index} - Category: {top_category}")
```

```
        NDA text 0 - Category: ('UNCAT', 0.9713473320007324)
        NDA text 1 - Category: ('UNCAT', 0.9999927282333374)
        NDA text 2 - Category: ('UNCAT', 0.9999996423721313)
        NDA text 2 - Category: ('ISSUE', 0.9057997465133667)
        NDA text 2 - Category: ('UNCAT', 0.996208906173706)
        NDA text 2 - Category: ('CONCLUSION', 0.8959798812866211)
        NDA text 3 - Category: ('UNCAT', 0.9696993231773376)
        NDA text 3 - Category: ('CONCLUSION', 0.9830033183097839)
        NDA text 4 - Category: ('UNCAT', 0.9988866448402405)
        NDA text 4 - Category: ('UNCAT', 0.6295406818389893)
        NDA text 5 - Category: ('CONCLUSION', 0.7646313905715942)
```

RASA-NLU

https://rasa.com/docs/rasa/nlu/about/

Rasa is an open-source conversational artificial intelligence (AI) framework. The first piece of a Rasa assistant is an NLU (Natural Language Understanding) model.

Rasa NLU is an open-source natural language processing tool for intent classification, response retrieval and entity extraction ⇒ Trainable model to text categoriser and named entity recogniser (NER).

Tutorial notebook: https://github.com/victor-roris/NLPlearning/blob/master/NER/RasaNLU.ipynb

We applied RASA-NLU in the projects:

- **NDA-INTERPRETER**
  - DESCRIPTION: https://docs.google.com/presentation/d/1-Gq6c7mSytNqBQ38jp4g-Tg-JZ3CSvRGMx-wY0-LU40/edit?usp=sharing
  - CODE: https://gitlab.dataspartan.com/victor/nda-interpreter-api/tree/feature/ndainterpreter/src/app/service

- **ORDER CHAT**
  - CODE: https://gitlab.dataspartan.com/victor/poc-orderchatbot

Text classification

Text classification is the process of assigning tags or categories to text according to its content [link].

Broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Text classification can be:

- **Binary**: it classifies text in one (and only one) of two categories (e.g., sentiment analysis: good or bad [link])
- **MultiClass**: it classifies text in one (and only one) of more than two categories (e.g, classify a newspaper texts by its category [link])
- **MultiLabel**: it classifies text in zero, one or more categories at same time (e.g, classify the plot text of a movie in its possible genres [link])

Word Embeddings (I)

http://hunterheidenreich.com/blog/intro-to-word-embeddings/

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

Different ways of calculate this embeddings:

- **Basic:** One hot encoding, TF-IDF, Co-Occurrence Matrix
- **Neural Probabilistic Models** (This typically takes a lot of data and can be very slow. Very good embeddings for the text data):
  - Word embeddings: ie the embedding layer of a model: Word2Vec, Glove
  - Library that provides word vectors: fastText

Word Embeddings (II)

Traditional word vectors are shallow representations used as previous knowledge in the first layer of the model. The rest of the network still needs to be trained from scratch for a new target task.

NLP Models with word embeddings (text classification):

- **Multi-hot arrays text embeddings (CountVectorizer [info]): Keras Model, Keras Model 2**
- **TFIDF vectorization [info]: LogisticRegression Model, RandomForest_Model, XGBoos_Model**
- **CountVectorizer & TFIDF vectorization:** NaiveBayes_Model, SVM_Model, LogisticRegression_Model
- **Word2Vec [info]: LogisticRegression_Model, Keras_Model**
- **Doc2Vec [info]: LogisticRegression_Model**
- **FastText:** FastText

*"It only seems to be a question of time until **pretrained word embeddings** will be dethroned and replaced by **pretrained language models** in the toolbox of every NLP practitioner" [Sebastian Ruder]*

PreTrained Models (I)

https://towardsdatascience.com/from-word-embeddings-to-pretrained-language-models-a-new-age-in-nlp-part-2-e9af9a0bdcd9

Traditional word vectors are shallow representations (a single layer of weights, known as embeddings). They only incorporate previous knowledge in the first layer of the model. The rest of the network still needs to be trained from scratch for a new target task.

Word embeddings are useful in only capturing semantic meanings of words but we also need to understand higher level concepts like anaphora, long-term dependencies, agreement, negation, and many more. These word embeddings are not context-specific — they are learned based on word concurrency but not sequential context (ex. “apple” word refer to very different things (fruit or company) but they would still share the same word embedding vector.

Deep Training

The standard way of conducting NLP projects has been — word embeddings pre-trained on large amounts of unlabeled data via algorithms such as word2vec and GloVe are used to initialize the first layer of a neural network, the rest of which is then trained on data of a particular task. Most datasets for text classification (or any other supervised NLP tasks) are rather small. This makes it very difficult to train deep neural networks, as they would tend to overfit on these small training data sets and not generalize well in practice.

Current state-of-art models for supervised NLP tasks are models pre-trained on language modeling (which is an unsupervised task), and then fine tuned (supervised) with labeled data specific to a task. All these approaches allow us to pre-train an unsupervised language model on large corpus of data such as all wikipedia articles, and then fine-tune these pre-trained models on downstream tasks.

Pre-Trained Models (II)

https://medium.com/@adriensieg/from-pre-trained-word-embeddings-to-pre-trained-language-models-focus-on-bert-343815627598

Language modeling is the task of assigning a probability distribution over sequences of words that matches the distribution of a language. Although it sounds formidable, l**anguage modeling** (i.e. ELMo, BERT, GPT) is essentially just predicting words in a blank. More formally, given a context, a language model predicts the probability of a word occurring in that context.

Static vs. Dynamic

- **Static Word Embeddings** fail to **capture polysemy**. They generate the **same embedding** for the **same word** in **different contexts**. ### Contextualized words embeddings aim at **capturing word semantics** in **different contexts** to address the **issue of polysemous** and the **context-dependent nature of words**.
- **Static Word Embeddings could only leverage off the vector outputs from unsupervised models for downstream tasks** — not the unsupervised models themselves.They were mostly **shallow models** to begin with and were often discarded after training (e.g. word2vec, Glove) ### The output of **Contextualized (Dynamic) Word Embedding** training is **the trained model and vectors** — not just vectors.
- Traditional word vectors are **shallow representations** (a single layer of weights, known as embeddings). They only **incorporate previous knowledge in the first layer of the model**. The rest of the network still needs to be trained from scratch for a new target task. They **fail to capture higher-level information** that might be even more useful. Word embeddings are useful in **only capturing semantic meanings of words** but we also need to understand higher level concepts like **anaphora**, **long-term dependencies**, agreement, negation, and many more.

Pre-Trained Models (III)

https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8

- UMLFit: first proposal of Pre-trained models
- ELMO: google proposal
- BERT: Evolution of pre-trained models with bidirectional training using maskering ([MASK]) of the pre-trained data (qualitative improvement)
- RoBERTA: Improve over BERT by Facebook
- XLNet: Use the BERT idea and use other bidirectional training strategy
- DistilBERT: Less efficiency than BERT but faster .
- GPT2

So which one to use?
If you really need a faster inference speed but can compromise few-% on prediction metrics, DistilBERT is a starting reasonable choice, however, if you are looking for the best prediction metrics, you'll be better off with Facebook's RoBERTa.

Theoretically, XLNet's permutation based training should handle dependencies well, and might work better in longer-run.

However, Google's BERT does serve a good baseline to work with and if you don't have any of the above critical needs, you can keep your systems running with BERT.

ULMFit

http://nlp.fast.ai/
https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts

ULMFiT is a model training technique consisting of three stages:

1. Training a language model on a large, general corpus
2. Taking that pretrained language model and fine tuning it on a specific corpus
3. Taking the fine tuned language model and training a classification model

We applied ULMFit in Lawbite project: https://gitlab.dataspartan.com/dataspartan/lawbite/development/lawbite-notebooks

Review new version: MultiFiT, a novel method based on ULMFiT.

BERT

paper, github

BERT (Bidirectional Encoder Representations from **Transformers**)

Using BERT has two stages: *Pre-training* and *fine-tuning*.

- Pre-training is fairly expensive (four days on 4 to 16 Cloud TPUs), but is a one-time procedure for each language (current models are English-only, but multilingual models will be released in the near future). We are releasing a number of pre-trained models from the paper which were pre-trained at Google. Most NLP researchers will never need to pre-train their own model from scratch.
- Fine-tuning is inexpensive. All of the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model. SQuAD, for example, can be trained in around 30 minutes on a single Cloud TPU to achieve a Dev F1 score of 91.0%, which is the single system state-of-the-art.

Examples of use:

- Using the original Google BERT scripts: notebook, scripts

  Complex. It can be reusable but it is difficult to understand.

- Using Hugging Face (explained latter): notebook, paconathan's_notebook, other_implementation

  It is bit easier because Hugging Face library and, of course, Paco Nathan's explanation.

- Microsoft Azure notebooks: github

Transformers

https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. The Encoder takes the input sequence and maps it into a **higher dimensional space (n-dimensional vector)**. That abstract vector is fed into the Decoder which turns it into an output sequence.

Imagine the Encoder and Decoder as human translators who can speak only two languages. Their first language is their mother tongue, which differs between both of them (e.g. German and French) and their second language an imaginary one they have in common. To translate German into French, the Encoder converts the German sentence into the other language it knows, namely the imaginary language. Since the Decoder is able to read that imaginary language, it can now translates from that language into French. Together, the model (consisting of Encoder and Decoder) can translate German into French!

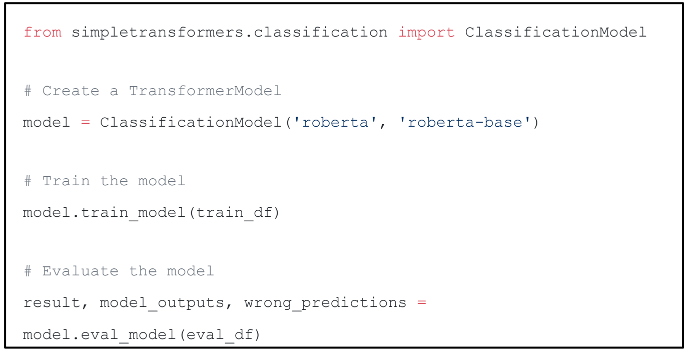
Transformer uses something called 'Attention' The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important.

Detailed explanation: part1 part2 paper

Transformers Libraries

- Hugging face: The Hugging Face library provides state-of-the-art general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch. The library includes architecture modifications designed for text classification, token classification, question answering, next sentence prediction, etc. Using these pre-built classes simplifies the process of modifying architectures for your purposes.

- Simple transformers: The technical barriers that need to be overcome in order to adapt Transformers to specific tasks are non-trivial and may even be rather discouraging. The idea was to make it as simple as possible, which means abstracting away a lot of the implementational and technical details.
  The Simple Transformers library was written so that a Transformer model can be initialized, trained on a given dataset, and evaluated on a given dataset, in just 3 lines of code!



```
from simpletransformers.classification import ClassificationModel

# Create a TransformerModel
model = ClassificationModel('roberta', 'roberta-base')

# Train the model
model.train_model(train_df)

# Evaluate the model
result, model_outputs, wrong_predictions =
model.eval_model(eval_df)
```

https://towardsdatascience.com/simple-transformers-introducing-the-easiest-bert-roberta-xlnet-and-xlm-library-58bf8c59b2a3
https://github.com/ThilinaRajapakse/simpletransformers

Transformers Libraries

KTRAIN

- KTRAIN:
  - TEXT_MULTICLASS CLASIFICATION
  - TEXT_MULTILABEL CLASIFICATION

Spacy transformers

https://github.com/explosion/spacy-transformers

`spacy-transformers` provides spaCy model pipelines that wrap **Hugging Face's transformers** package. It allows access to the BERT, RoBERTa, XLNet and GPT-2 directly.

- https://github.com/DerwenAI/spaCy_tuTorial/blob/master/spaCy_transformers_demo.ipynb
- https://github.com/victor-roris/NLPlearning/blob/master/Transformers/Spacy_PyTorch_Transformers.ipynb
- https://github.com/victor-roris/NLPlearning/blob/master/Transformers/Spacy_PyTorch_Transformers_Demo.ipynb
- https://github.com/victor-roris/NLPlearning/blob/master/Transformers/Spacy_Transformers.ipynb
- (Train Text Categorizer) https://github.com/victor-roris/NLPlearning/blob/master/Transformers/Spacy_Transformers_Train_Text_Categorizer.ipynb
- (Train Bert Model) https://github.com/victor-roris/NLPlearning/blob/master/text_classification/BERT_HuggingFace_Fine_Tuning_Sentence_Classification-Multiclass.ipynb

Interpretability - Understanding Model Predictions (I)

Prefer solutions that are interpretable and understandable ⇒ allows to validate and improve work

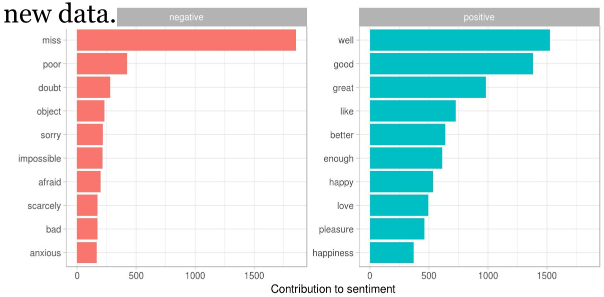
Interpretability helps:

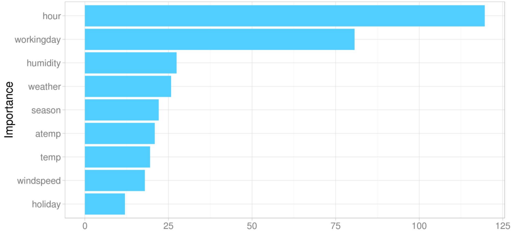
- *Identify and mitigate bias*
- *Accounting for the context of the problem:* helps you to understand the factors that are (not) included in the model
- *Improving generalisation and performance*: the combination of solid data, model and problem understanding is necessary to have a solution that performs better.
- *Ethical and legal reasons:* ensure it is e.g. not discriminatory or violating any laws

Interpretability - Understanding Model Predictions (II)

Feature importance (what the model is learning)

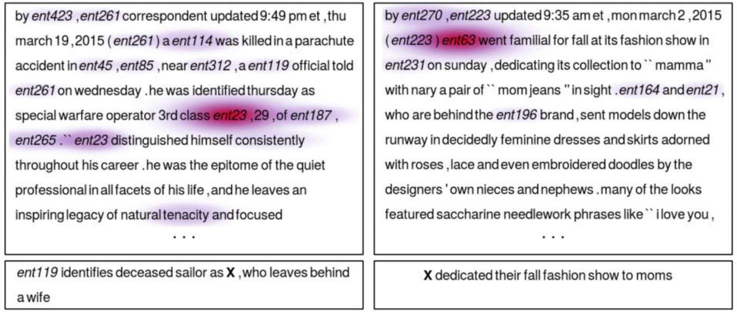
- When building a text classifier you can plot the most important features and verify whether the model is overfitting on noise. If the most important words do not correspond to your intuition (e.g. names or stopwords), it probably means that the model is fitting to noise in the dataset and it won't perform well on new data.





- By looking at the feature importance, you can identify what the model is learning. As a lot of importance in this model is put into time of the day, it might be worthwhile to incorporate additional time-based features.

- Attention-based methods are typically used with sequential data (e.g. text data). In addition to the normal weights of the network, attention weights are trained that act as 'input gates'. These attention weights determine how much each of the different elements in the final network output. Besides interpretability, attention within the context of the e.g. text-based question-answering also leads to better results as the network is able to 'focus' its attention.



by *ent423* ,*ent261* correspondent updated 9:49 pm et ,thu march 19 ,2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* ,*ent85* ,near *ent312* ,a *ent119* official told *ent261* on wednesday .he was identified thursday as special warfare operator 3rd class *ent23* ,29 ,of *ent187* , *ent265* .`` *ent23* distinguished himself consistently throughout his career .he was the epitome of the quiet professional in all facets of his life ,and he leaves an inspiring legacy of natural tenacity and focused

. . .

ent119 identifies deceased sailor as **X** ,who leaves behind a wife

by *ent270* ,*ent223* updated 9:35 am et ,mon march 2 ,2015 (*ent223*) *ent63* went familial for fall at its fashion show in *ent231* on sunday ,dedicating its collection to `` mamma " with nary a pair of `` mom jeans " in sight .*ent164* and *ent21* , who are behind the *ent196* brand ,sent models down the runway in decidedly feminine dresses and skirts adorned with roses ,lace and even embroidered doodles by the designers ' own nieces and nephews .many of the looks featured saccharine needlework phrases like `` i love you ,

. . .

X dedicated their fall fashion show to moms

Interpretability - Understanding Model Predictions (III)

https://towardsdatascience.com/3-ways-to-interpretate-your-nlp-model-to-management-and-customer-5428bc07ce15

Interpretability

- **Intrinsic:** We do not need to train another model to explain the target. For example, it is using decision tree or liner model.
- **Post hoc:** The model belongs to black-box model which we need to use another model to interpret it.

Approach

- **Model-specific:** Some tools are limited to specific model such as linear model and neural network model.
- **Model-agnostic:** On the other hand, some tools able to explain any model by building write-box model.

Level

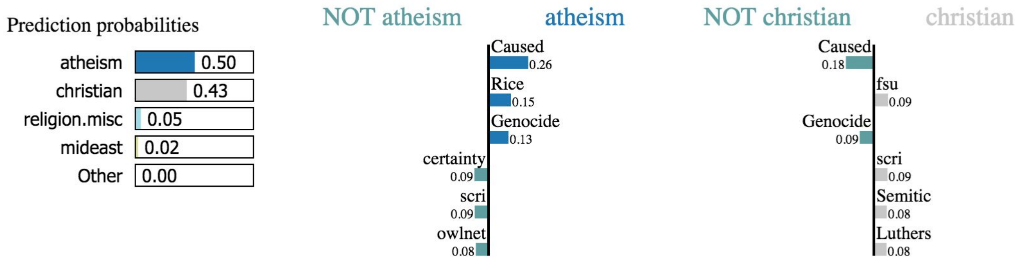
- **Global:** Explain the overall model such as feature weight. This one give you a in general model behavior
- **Local:** Explain the specific prediction result.

Interpretability - LIME

https://github.com/marcotcr/lime

The technique attempts to understand the model by perturbing the input of data samples and understanding how the predictions change. In the context of text classification, this means that some of the words are e.g. replaced, to determine which elements of the input impact the predictions.

The output of LIME is a list of explanations, reflecting the contribution of each feature to the prediction of a data sample ⇒ local interpretability, and it also allows to determine which feature changes will have most impact on

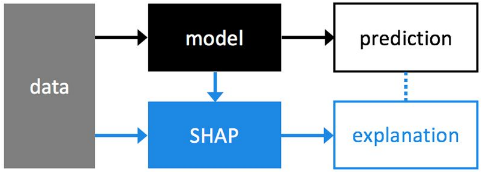


Tutorial Notebook: Notebook1, Notebook2

It was applied to interpreter
the Lawbite NLP Model (ULMFit) : Notebook

Interpretability - SHAP

https://github.com/slundberg/shap
https://shap.readthedocs.io/en/latest/



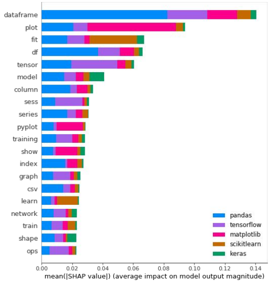
SHAP (SHapley Additive exPlanations) is a unified approach to explain the output of any machine learning model.

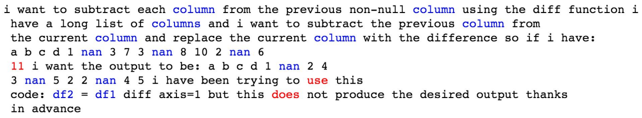
ML models supported are: XGBoost, LightGBM, CatBoost, Pyspark and most tree-based scikit-learn models.

SHAP includes **multiple algorithms**. You may check out paper for more detail on LIME, DeepLIFT, Sharpley value calculations.

Tutorial Notebook: Notebook

Google example: BlogEntry, Video





```
i want to subtract each column from the previous non-null column using the diff function i
 have a long list of columns and i want to subtract the previous column from
 the current column and replace the current column with the difference so if i have:
 a b c d 1 nan 3 7 3 nan 8 10 2 nan 6
 11 i want the output to be: a b c d 1 nan 2 4
 3 nan 5 2 2 nan 4 5 i have been trying to use this
 code: df2 = df1 diff axis=1 but this does not produce the desired output thanks
 in advance
```

Interpretability - Eli5

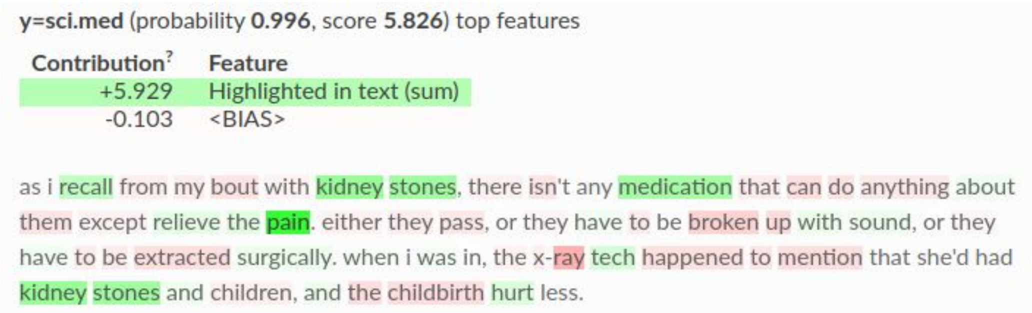
https://github.com/TeamHG-Memex/eli5
https://eli5.readthedocs.io/en/latest/index.html

ELI5 provides both global model interpretation and local model interpretation.

The global interpretation works for: scikit-learn, Keras, xgboost, LightGBM, CatBoost, lightning, sklearn-crfsuite.

For local model interpretation, ELI5 use **LIME** algorithm. The display format is different from LIME but using same idea.

Tutorial Notebook: Notebook



y=sci.med (probability **0.996**, score **5.826**) top features

Contribution?	Feature
+5.929	Highlighted in text (sum)
-0.103	<BIAS>

as i recall from my bout with kidney stones, there isn't any medication that can do anything about them except relieve the pain. either they pass, or they have to be broken up with sound, or they have to be extracted surgically. when i was in, the x-ray tech happened to mention that she'd had kidney stones and children, and the childbirth hurt less.

To review

[other model]

https://openai.com/blog/better-language-models/
https://medium.com/@ageitgey/deepfaking-the-news-with-nlp-and-transformer-models-5e057ebd697d

[interpretation]

https://allennlp.org/interpret - https://demo.allennlp.org/masked-lm