Classify Claims

Homeproject in Data & Knowledge Engineering

17.01.2023

The General Idea

Only consider claim texts

- Bias vs. Performance
- Problems with the metadata supported the decision

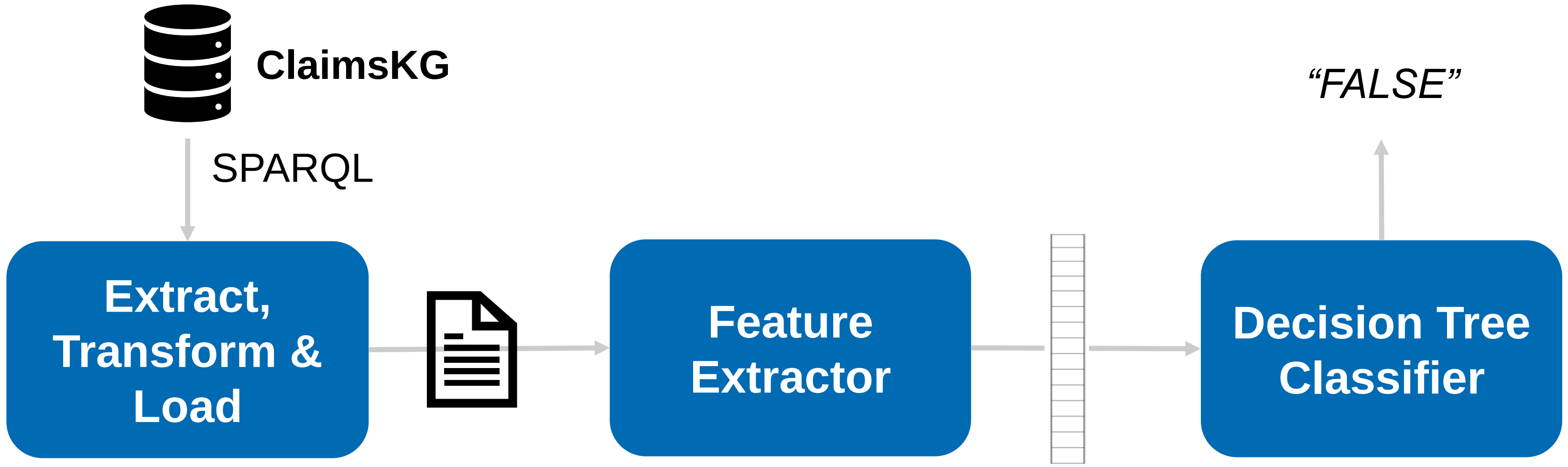
Feature Engineering

- Extract sentiment, emotional and textual features from each claim
- Dense vector representations

Train a ML model on extracted features

- Interpretable model like Decision Tree Classifier

Overview Architecture and Modules



External modules:

- Language Detector
- Translator (google)

External modules:

- Sentiment Analyser (nltk vader)
- Emotion Extractor (text2emotion & NRCLex)

External modules:

- Scikit-Learn

Data Acquisition

Heinrich Heine
Universität Düsseldorf

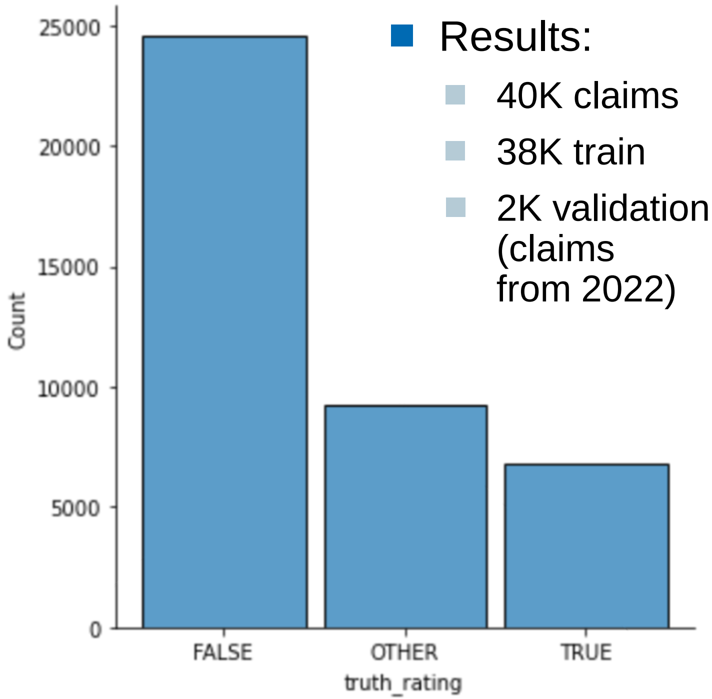
```
PREFIX itsrdf:<https://www.w3.org/2005/11/its/rdf#>
    PREFIX schema:<http://schema.org/>
    PREFIX dbr:<http://dbpedia.org/resource/>
    SELECT ?claim ?text ?date ?ratval
    WHERE {
                    {
                            SELECT ?claim ?text ?date ?ratval
                            WHERE {
                            ?review a schema:ClaimReview .
                            ?review schema:reviewRating ?rating .
                            ?rating schema:alternateName ?ratval .
                            ?review schema:itemReviewed ?claim .
                            ?claim schema:text ?text .
                ?review schema:datePublished ?date .
                FILTER regex(?ratval , "(^FALSE|TRUE|OTHER)")
                            } ORDER BY ?claim
                    }
            }
        LIMIT 10000 OFFSET
```

- Missing/changing parts are appended before querying endpoint
- Takes ~12min to query all data from ClaimsKG

```
    PREFIX itsrdf:<https://www.w3.org/2005/11/its/rdf#>
PREFIX schema:<http://schema.org/>
PREFIX dbr:<http://dbpedia.org/resource/>
SELECT ?claim ?text
WHERE{
               {
                      SELECT ?claim ?text
                      WHERE {
                      ?claim a schema:CreativeWork .
           ?claim schema:text ?text .
                      }
               } FILTER regex(?claim ,
```

Data Preprocessing & Cleaning

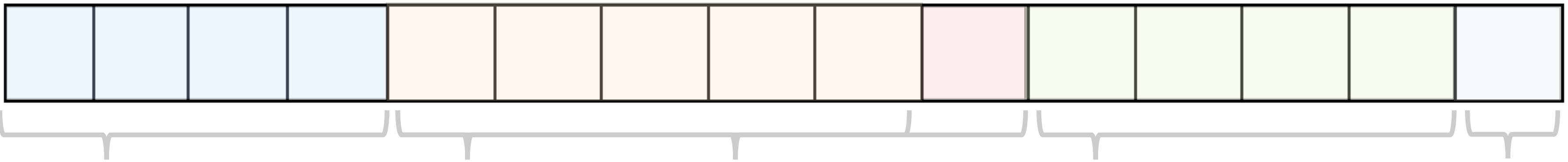
- Remove duplicates
  - ID
  - Claim text (inconsistent rating)
- Removed very short claims
  - Less than 3 token
  - Likely not representative
- Removed non-English claims that are “FALSE”
- Translated “TRUE” and “OTHER” claims into English
  - Account for underrepresented classes
- Claims from 2022 are used as validation data
  - Test data excluded!



- Results:
  - 40K claims
  - 38K train
  - 2K validation (claims from 2022)

A Closer Look on The Features

Heinrich Heine
Universität Düsseldorf



- Sentiment
  - Positive
  - Neutral
  - Negative
  - Compound

- Emotions
  - Anger
  - Fear
  - Happiness
  - Sadness
  - Surprise

- Emotion density
  - Inverse number of different emotions
  - To normalize inconsistent output shape

- Textual features
  - Tokenize claim
  - # numeric token
  - # exclamation marks
  - # question marks
  - # token that contain an uppercase letter (proxy for entities)
  - All normalized by # of claim token

- Claim Length
  - Total number of token in the claim

Some Results

- Model: Decision Tree Classifier
  - Split by entropy, minimal 30 samples in leaf
- Most important features:
  - Number of capital words
  - Length of claim
  - Low importance on sentiment and emotion
- Combine features & models
  - Improved results for nearly all classifier (max Accuracy: 55%)
  - Classifier ensemble didn’t improve performance

Results on validation set:

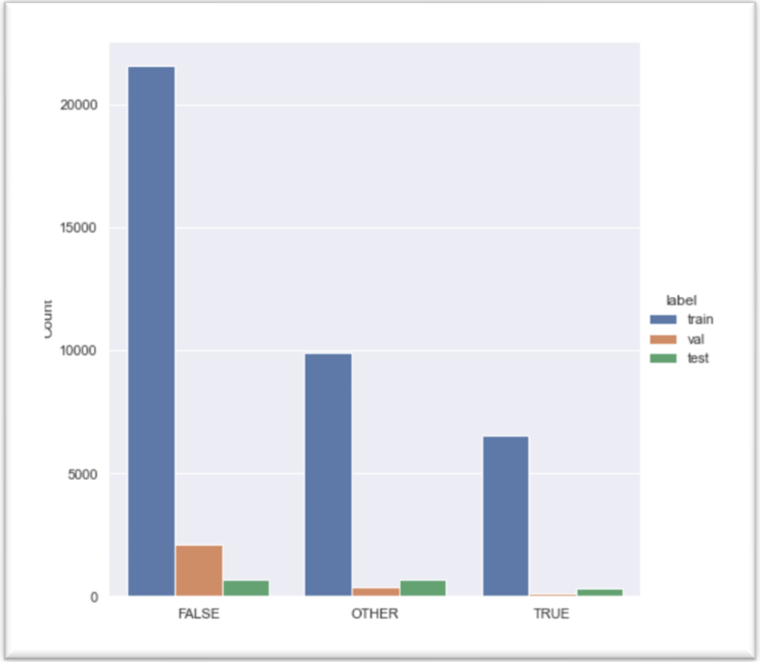
	precision	recall	f1-score	support
FALSE	0.83	0.90	0.86	2119
OTHER	0.25	0.15	0.19	396
TRUE	0.14	0.11	0.12	110
accuracy			0.76	2625
macro avg	0.41	0.39	0.39	2625
weighted avg	0.71	0.76	0.73	2625

Results on test set:

	precision	recall	f1-score	support
FALSE	0.46	0.90	0.61	700
NEITHER	0.59	0.20	0.29	679
TRUE	0.36	0.11	0.17	301
accuracy			0.48	1680
macro avg	0.47	0.40	0.36	1680
weighted avg	0.50	0.48	0.40	1680

Insights & Outlook

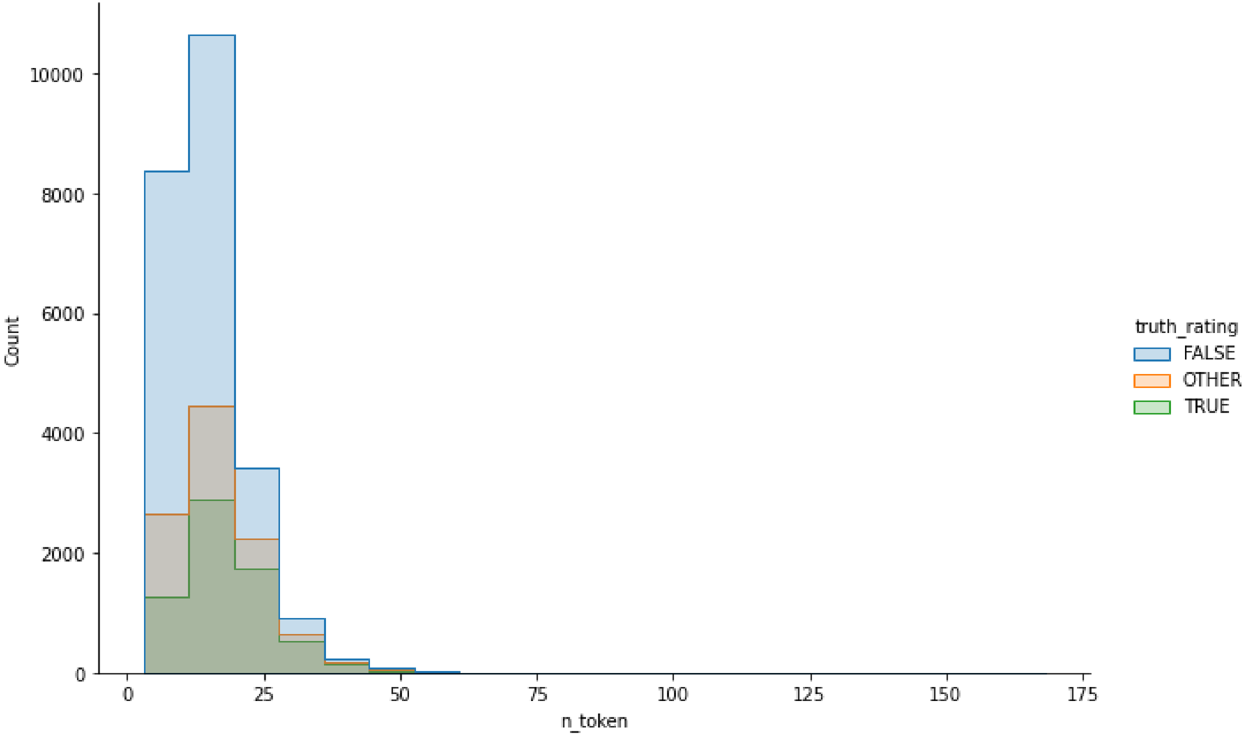
- Distribution shift
  - Leads to bad performance on test set
  - Learning: representative validation set?
- Accuracy is not the best measure for clf
  - Best model didn’t predict any “TRUE”
- Often classifiers showed similar performance
  - Similar for other features (sparse & dense)
- Future: Pre-trained LMs
  - Could massively improve on performance (but are costly)
  - Fine-tune on claims
  - Get contextualized embeddings



Github Repository: https://github.com/niruc100/classify_claims

Thank you!

Distribution of Labels by Claim Length



Data Augmentation

Heinrich Heine Universität Düsseldorf

- Tried to rebalance the amount of FALSE, TRUE and OTHER claims by augmenting true and other claims
- Augmentation by translation: Translate sentence into another language and back to retrieve the same meaning but different words
  - Translated to Chinese and Indian
  - Removed all duplicates again (~600 of 10K augmented claims were actually word by word translations)
  - Slow and expensive (network cost)
- In the end helped for first smaller data but more “real” claims were better than augmented claims

Problems with ClaimsKG

hhu
Heinrich Heine
Universität Düsseldorf

- SPARQL endpoint has 10K limit
  - Requires workaround with multiple requests
  - DBPedia allows 40K per query
- The shown schema and especially statistics on the website are not true
- Problems in data retrieved from ClaimsKG
  - language is labeled English but is not always English
  - Multiple IDs (up to an ID that is repeated 976 times)
  - Date published from the claim is only present in half the claims
  - Empty claim text (especially in africheck)
  - Claim author can be the instance or the review author

NRCLex - The Library that detects emotions

NRCLex(or NRCLexicon) is an MIT-approved PyPI project by Mark M. Bailey which predicts the sentiments and emotion of a given text. The package contains approximately 27,000 words and is based on the National Research Council Canada (NRC) affect lexicon and the NLTK library's WordNet synonym sets.

Emotional affects measured include the following:

1. fear
2. anger
3. anticipation
4. trust
5. surprise
6. positive
7. negative
8. sadness
9. disgust
10. joy

Example:

`from nrclex import NRCLex`

`text = 'hate'`

`emotion = NRCLex(text)`

`print(emotion.raw_emotion_scores)` #Return raw emotional counts(for each emotion).

Link to the official project website: https://pypi.org/project/NRCLex/

- Source: https://www.kaggle.com/getting-started/196520
- Library: https://pypi.org/project/NRCLex/

Heinrich Heine Universität Düsseldorf

Using NLTK's Pre-Trained Sentiment Analyzer

NLTK already has a built-in, pretrained sentiment analyzer called VADER (**V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner).

Since VADER is pretrained, you can get results more quickly than with many other analyzers. However, VADER is best suited for language used in social media, like short sentences with some slang and abbreviations. It's less accurate when rating longer, structured sentences, but it's often a good launching point.

To use VADER, first create an instance of `nltk.sentiment.SentimentIntensityAnalyzer`, then use `.polarity_scores()` on a raw string:

Python >>>

```
>>> from nltk.sentiment import SentimentIntensityAnalyzer
>>> sia = SentimentIntensityAnalyzer()
>>> sia.polarity_scores("Wow, NLTK is really powerful!")
{'neg': 0.0, 'neu': 0.295, 'pos': 0.705, 'compound': 0.8012}
```

- Source: https://realpython.com/python-nltk-sentiment-analysis/#:~:text=Sentiment%20analysis%20is%20the%20practice,obtain%20insights%20from%20linguistic%20data.
- Package: https://www.nltk.org/api/nltk.sentiment.vader.html

Heinrich Heine Universität Düsseldorf

Text2Emotion

```
text = "I was asked to sign a third party contract a week out from
stay. If it wasn't an 8 person group that took a lot of wrangling I
would have cancelled the booking straight away. Bathrooms - there are
no stand alone bathrooms. Please consider this - you have to clear
out the main bedroom to use that bathroom. Other option is you walk
through a different bedroom to get to its en-suite. Signs all over
the apartment - there are signs everywhere - some helpful - some
telling you rules. Perhaps some people like this but It negatively
affected our enjoyment of the accommodation. Stairs - lots of them -
some had slightly bending wood which caused a minor injury."
```

Now we have to call the get_emotion() function using the above-defined **text** parameter.

```
#Call to the function
te.get_emotion(text)

#The output we received,
{'Angry': 0.12, 'Fear': 0.42, 'Happy': 0.04, 'Sad': 0.33, 'Surprise':
0.08}
```

- Source:https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153
- Package: https://pypi.org/project/text2emotion/

Decision Tree Classifier