Classifying Propositional Content in annotated Argumentative Discourse Units

Processamento de Linguagem Natural

FEUP

21/04/2022

Clara Gadelho, up201806309
Flávia Carvalhido, up201806857

Table of contents

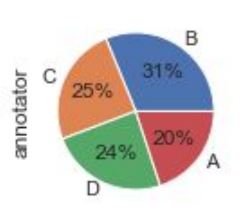
Data Analysis

We had access to two data files:

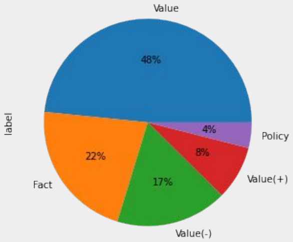
- One with the content of each ADU and respective assigned class (16742 rows)
- Another with metadata and full article from which ADUs were obtained (373 rows)

Main challenges:

- The text is in Portuguese
- There's a significant class imbalance
- The annotators often disagree on the assigned label (5510 rows)



label	
Value	8102
Fact	3663
Value(-)	2900
Value(+)	1411
Policy	667



	article_id	annotator	node	ranges	tokens	label
1654	5d04c505896a7fea06a0fabc	A	0	[[0, 104]]	Em dezembro do ano passado Fernando Medina ava...	Value
1669	5d04c505896a7fea06a0fabc	B	0	[[0, 104]]	Em dezembro do ano passado Fernando Medina ava...	Fact
4654	5cf47065896a7fea060065b5	C	1	[[0, 108]]	As instituições de ensino superior portuguesas...	Value(+)
4691	5cf47065896a7fea060065b5	D	0	[[0, 108]]	As instituições de ensino superior portuguesas...	Value

Data preprocessing

We ended focusing on the main dataset, not including data from the articles.

Char normalization

Removal of accentuation, special chars, upper case and stopwords.

Sampling

As the dataset is imbalanced, we used sampling techniques to try to improve our results.
Algorithms used: Random Undersampling, NearMiss Undersampling, Condensed Nearest Neighbor Rule Undersampling and SMOTE.

Building a Corpus

This was challenging since we add to process portuguese text
Techniques/algorithms used: SnowballStemmer, RSLPStemmer, NLPyPort Lemmatizer, Stanza preprocessing and SpaCy preprocessing

Word Representation

We tried: CountVectorizer and TF-IDF

Data preprocessing

Dealing with ambiguity

When feeding our classifiers this ambiguous data, it will make it harder for them to achieve good results. Our idea was to list every ADU which had multiple labels assigned, then removing them following this logic:

- if it was only found by 2 or 3 annotators and they all disagree on the label, then all the entries of that ADU are removed
- if it is found by 3 annotators and 2 of them agree, then only the different one is removed.

```
def dropAmbiguousADUs(df):
    aux=df[df.duplicated(['ranges', 'article_id'], keep=False)].sort_values(by=['article_id', 'ranges'])
    aux=aux.drop_duplicates(subset=['article_id','ranges','label'],keep=False).sort_values(by=['article_id', 'ranges'])
    aux.index.name = 'id'
    tabuIndexes=aux.index.values
    cleandf=df.drop(tabuIndexes)
    return cleandf
```

Classification task

The task given consists of a multiclass classification problem: attributing a label (Value, Fact, Value(-), Value(+), Policy) to Argumentative Discourse Units extracted from portuguese articles.

We tried to test a wide variety of classifiers, as well as their respective parameters:

- **Logistic Regression**
  - Solver: saga, lbfgs, sag, newton-cg (with all allowed Penalties)
  - Penalty: None, l1, l2, elasticnet (with different l1_ratio)
- **Decision Tree**
  - Criterion: gini, entropy; Max Depth: None, 25 and 50; Max Features: auto
- **Random Forest**
  - Criterion: gini, entropy; Max Depth: None, 25 and 50; Max Features: auto
- **SVC**
  - Different C values
  - Kernel: linear, poly (gamma auto and scale), sigmoid (gamma auto and scale), rbf (gamma auto and scale), precomputed
  - Different Degrees and Coef0
- **Perceptron**
  - Penalty: l1, l2, elasticnet (with different l1_ratio)

Classification task

Ensemble Learning

We tried the following classifiers:

- **Bagging** (using Decision Trees as Base Estimator)
- **Extra Trees** (with criteria Gini and Entropy)
- **Hist Gradient Boosting** (with loss Auto, Categorical Cross Entropy and Deviance)
- **Ada Boost** (using Decision Trees as Base Estimator)
- **Voting Classifier**
  - Combinations of: LogisticRegression, RandomForest, Naive Bayes and Decision Trees

To help us with parameter choice, we used GridSearch

Evaluation

Metrics Used

The metrics we used to evaluate the performance were:

- Precision
- Recall
- F1-score
- Accuracy
- AUC score (adapted to consider multiple classes)
- Confusion Matrix

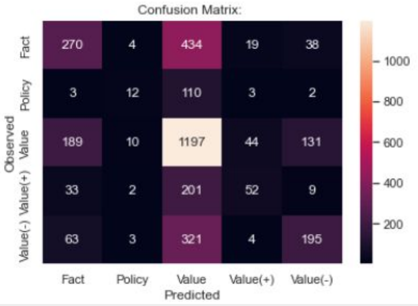
Since this was a multiclass problem we also had to take into account the macro and micro averages of the metrics.
We compared all our results with our baseline: a simple preprocessing with Multinomial Naive Bayes Classification.

```
Report:
              precision    recall  f1-score   support

        Fact       0.48      0.35      0.41       765
      Policy       0.39      0.09      0.15       130
       Value       0.53      0.76      0.62      1571
    Value(+)       0.43      0.18      0.25       297
    Value(-)       0.52      0.33      0.41       586

    accuracy                           0.52      3349
   macro avg       0.47      0.34      0.37      3349
weighted avg       0.50      0.52      0.48      3349


AUC Score:
{'Fact': 0.6207430340557276, 'Policy': 0.5432026190646881, 'Value':
0.5811925084794063, 'Value(+)': 0.5760741976338307, 'Value(-)': 0.6
338089626574467}
```



Evaluation
Preprocessing results

Sampling

The results obtained were very underwhelming: NearMiss had the worse results but all were worse than no sampling

Random Undersampling

```
Report:
              precision    recall  f1-score   support
        Fact       0.40      0.53      0.46       765
      Policy       0.44      0.17      0.24       130
       Value       0.58      0.52      0.55      1571
    Value(+)       0.40      0.31      0.35       297
    Value(-)       0.43      0.49      0.46       586

    accuracy                           0.48      3349
   macro avg       0.45      0.40      0.41      3349
weighted avg       0.49      0.48      0.48      3349

AUC Score:
'Value(-)': 0.6751320780820175, 'Value': 0.5965066349519805,
'Policy': 0.5802662078524148, 'Value(+)': 0.6310820083755864,
'Fact': 0.6477597179222566
```

Confusion Matrix:

Near Miss Undersampling

```
Report:
              precision    recall  f1-score   support
        Fact       0.39      0.15      0.21       765
      Policy       0.06      0.92      0.11       130
       Value       0.57      0.07      0.13      1571
    Value(+)       0.18      0.29      0.22       297
    Value(-)       0.38      0.18      0.25       586

    accuracy                           0.16      3349
   macro avg       0.32      0.32      0.18      3349
weighted avg       0.44      0.16      0.18      3349

AUC Score:
'Value(-)': 0.5608436197979394, 'Value': 0.51241641421175,
'Policy': 0.6509200181613974, 'Value(+)': 0.5778944976192683,
'Fact': 0.5387598899208805
```

Confusion Matrix:

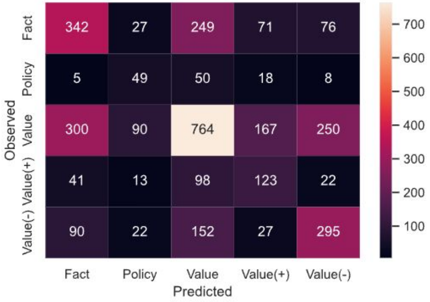
SMOTE

```
Report:
              precision    recall  f1-score   support
        Fact       0.44      0.45      0.44       765
      Policy       0.24      0.38      0.30       130
       Value       0.58      0.49      0.53      1571
    Value(+)       0.30      0.41      0.35       297
    Value(-)       0.45      0.50      0.48       586

    accuracy                           0.47      3349
   macro avg       0.40      0.45      0.42      3349
weighted avg       0.49      0.47      0.48      3349

AUC Score:
'Value(-)': 0.6872837557237953, 'Value': 0.5887702730665987,
'Policy': 0.6648517217482734, 'Value(+)': 0.6607076664416114,
'Fact': 0.6391640866873065}
```

Confusion Matrix:



Evaluation

Preprocessing results

Building the Corpus

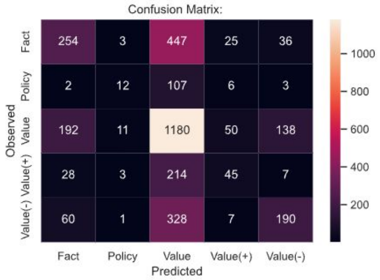
SnowballStemmer: the stemmer that produced the best results
RSLPStemmer: worse than Snowball
NLPyPort: very slow, it was impossible to test
Stanza and SpaCy preprocessing: easy and reliable but the results were underwhelming

RSLPStemmer

Report:

	precision	recall	f1-score	support
Fact	0.47	0.33	0.39	765
Policy	0.40	0.09	0.15	130
Value	0.52	0.75	0.61	1571
Value(+)	0.34	0.15	0.21	297
Value(-)	0.51	0.32	0.40	586
accuracy			0.50	3349
macro avg	0.45	0.33	0.35	3349
weighted avg	0.49	0.50	0.47	3349

AUC Score:
{'Value(-)': 0.6288189001666339, 'Value': 0.5673454965169457, 'Policy': 0.5433579468062227, 'Value(+)': 0.56134079987291, 'Fact': 0.6114465084279326}



SpaCy Preprocessing

Report:

	precision	recall	f1-score	support
Fact	0.63	0.45	0.52	600
Policy	0.89	0.08	0.14	106
Value	0.59	0.86	0.70	1346
Value(+)	0.58	0.16	0.25	237
Value(-)	0.61	0.39	0.48	457
accuracy			0.60	2746
macro avg	0.66	0.38	0.42	2746
weighted avg	0.61	0.60	0.56	2746

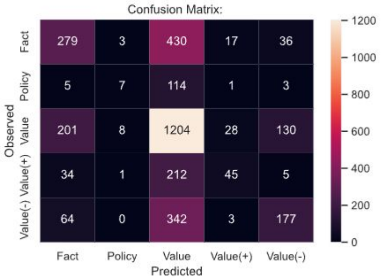
AUC Score:
{'Value(+)': 0.5726784419969965, 'Value': 0.6429776056039057, 'Policy': 0.5375464551172098, 'Value(-)': 0.6702835270578631, 'Fact': 0.6852213420316869}

Stanza Prepocessing

Report:

	precision	recall	f1-score	support
Fact	0.48	0.36	0.41	765
Policy	0.37	0.05	0.09	130
Value	0.52	0.77	0.62	1571
Value(+)	0.48	0.15	0.23	297
Value(-)	0.50	0.30	0.38	586
accuracy			0.51	3349
macro avg	0.47	0.33	0.35	3349
weighted avg	0.50	0.51	0.48	3349

AUC Score:
{'Value(-)': 0.6195363772127788, 'Value': 0.5744215136697983, 'Policy': 0.5250591440246613, 'Value(+)': 0.567730052821796, 'Fact': 0.6235294117647059}



Evaluation

Preprocessing results

Building the Corpus

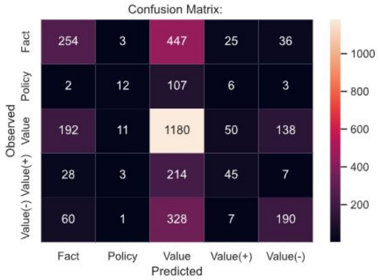
SnowballStemmer: the stemmer that produced the best results
RSLPStemmer: worse than Snowball
NLPyPort: very slow, it was impossible to test
Stanza and SpaCy preprocessing: easy and reliable but the results were underwhelming

RSLPStemmer

Report:

	precision	recall	f1-score	support
Fact	0.47	0.33	0.39	765
Policy	0.40	0.09	0.15	130
Value	0.52	0.75	0.61	1571
Value(+)	0.34	0.15	0.21	297
Value(-)	0.51	0.32	0.40	586
accuracy			0.50	3349
macro avg	0.45	0.33	0.35	3349
weighted avg	0.49	0.50	0.47	3349

AUC Score:
{'Value(-)': 0.6288189001666339, 'Value': 0.5673454965169457, 'Policy': 0.5433579468062227, 'Value(+)': 0.56134079987291, 'Fact': 0.6114465084279326}



SpaCy Preprocessing

Report:

	precision	recall	f1-score	support
Fact	0.63	0.45	0.52	600
Policy	0.89	0.08	0.14	106
Value	0.59	0.86	0.70	1346
Value(+)	0.58	0.16	0.25	237
Value(-)	0.61	0.39	0.48	457
accuracy			0.60	2746
macro avg	0.66	0.38	0.42	2746
weighted avg	0.61	0.60	0.56	2746

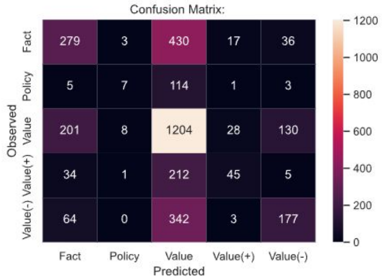
AUC Score:
{'Value(+)': 0.5726784419969965, 'Value': 0.6429776056039057, 'Policy': 0.5375464551172098, 'Value(-)': 0.6702835270578631, 'Fact': 0.6852213420316869}

Stanza Prepocessing

Report:

	precision	recall	f1-score	support
Fact	0.48	0.36	0.41	765
Policy	0.37	0.05	0.09	130
Value	0.52	0.77	0.62	1571
Value(+)	0.48	0.15	0.23	297
Value(-)	0.50	0.30	0.38	586
accuracy			0.51	3349
macro avg	0.47	0.33	0.35	3349
weighted avg	0.50	0.51	0.48	3349

AUC Score:
{'Value(-)': 0.6195363772127788, 'Value': 0.5744215136697983, 'Policy': 0.5250591440246613, 'Value(+)': 0.567730052821796, 'Fact': 0.6235294117647059}



Evaluation

Preprocessing results

Word Representation

Surprisingly enough we had better results with CountVectorizer

Annotator’s labelling divergence

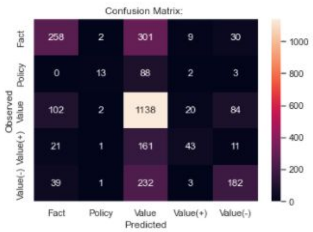
As expected, we had a huge improvement in performance when we used our version of the dataset without the label ambiguity.

Report:

	precision	recall	f1-score	support
Fact	0.61	0.43	0.51	600
Policy	0.68	0.12	0.21	106
Value	0.59	0.85	0.70	1346
Value(+)	0.56	0.18	0.27	237
Value(-)	0.59	0.40	0.47	457
accuracy			0.60	2746
macro avg	0.61	0.40	0.43	2746
weighted avg	0.60	0.60	0.56	2746

AUC Score:
{'Value': 0.6434483124601995, 'Policy': 0.5601843910806176, 'Fact': 0.677255358807083, 'Value(-)': 0.6711649187007024, 'Value(+)': 0.5839416917661818}

Cohen kappa Score:
0.3277465439583749

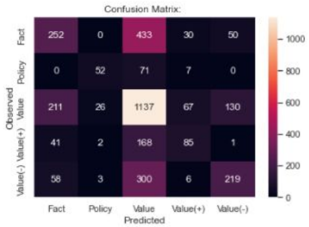


Report:

	precision	recall	f1-score	support
Fact	0.45	0.33	0.38	765
Policy	0.63	0.40	0.49	130
Value	0.54	0.72	0.62	1571
Value(+)	0.44	0.29	0.35	297
Value(-)	0.55	0.37	0.44	586
accuracy			0.52	3349
macro avg	0.52	0.42	0.46	3349
weighted avg	0.51	0.52	0.50	3349

AUC Score:
{'Value': 0.5885305870820889, 'Policy': 0.6951848400124262, 'Fact': 0.6047213622291021, 'Value(-)': 0.6541058156354262, 'Value(+)': 0.6250766732418108}

Cohen kappa Score:
0.25074610919241946



Evaluation

Classifier results

The best results we obtained were with Random Forest with criterion: entropy and max_depth:none

Report:

	precision	recall	f1-score	support
Fact	0.62	0.44	0.51	600
Policy	0.62	0.37	0.46	106
Value	0.64	0.84	0.73	1346
Value(+)	0.67	0.47	0.55	237
Value(-)	0.70	0.50	0.58	457
accuracy			0.65	2746
macro avg	0.65	0.52	0.57	2746
weighted avg	0.65	0.65	0.63	2746

```
AUC Score:
{'Value(+)': 0.7253263777825987, 'Value': 0.6948195712163022, 'Policy': 0.6794168096054889, 'Value(-)': 0.7271705703139265,
'Fact': 0.6805886921404163}
```

Confusion Matrix:

Conclusion

All in all, the task presented was very challenging and the results obtained could still be greatly improved.
The fact that the dataset was in portuguese forced us out of our comfort zone and to understand the progress of the development of tools for non-english text.

We were able to explore a wide variety of processing techniques and classifiers, gaining knowledge on this field of study.
As further improvement, we would have liked to include the data from the articles' as part of the classification and to try ULMFit.

We can safely say that the presented techniques in class have helped us understand much better all the technologies that were applied throughout. It is certain that our knowledge in NLP has increased immensely throughout the development of this project.

Questions?

References

Improving NLTK for Processing Portuguese

Portuguese Lemmatizers

Text Pre-Processing for Portuguese — Introduction to Cultural Analytics & Python

Evaluating Multi-Class Classifiers

Evaluation Metrics For Multi-class Classification

Introducing Metadata Enhanced ULMFiT

Undersampling and oversampling imbalanced data

Count Vectorizer vs TFIDF Vectorizer

Categorical Metadata Representation for Customized Text Classification