Legal Data Science & Informatics Project - WS2021

Name: Mahaut Gérard
Gloss ID: <tum_ldsi_21>

Summary

Annotated legal texts are necessary to build supervised machine learning models to support workers from the legal domain such as judges or attorneys. However, creating corpus of annotated legal texts requires legal domain experts, is time consuming, and human annotations fail to represent a universal ground truth. We propose a pipeline for automatic annotation of sentences from case decisions of the US Board of Veteran's Appeal (BVA). A BVA unlabeled dataset is used for training the Fasttext word embeddings and a BVA labeled dataset helps to train several multi-class (14) classification models after pre-processing steps including sentence segmentation, sentences normalization, tokenization, and sentence featurization.

Dataset Splitting

The dataset splitting was conducted using the "data_splitting" function from the "preprocessin.py" file. The function takes as parameter the dataset and the outcome type researched ("granted" or "denied"). For each type of outcome, the function first selects randomly (random.sample function) 14 document IDs. Second, in these 14 selected documents, the function selects 7 to be part of the test set while the 7 others are in the dev set. Consequently, we ran 2 times the function, 1 for "denied" cases and 1 for "granted" cases. The detailed splitting of our labeled dataset was stored in the "curated_annotations_split.yml" file of my code. This file was read as a reference each time the training dataset was built. The test dataset was never used before the final evaluation of the model. Here is the list of the documents' IDs in test and dev sets. All other documents of the dataset can be considered as train set.

TEST	granted:	denied:
	- 61aea55c97ad59b4cfc41298	- 61aea57497ad59b4cfc413db
	- 61aea55c97ad59b4cfc4128e	- 61aea57497ad59b4cfc413d1
	- 61aea55f97ad59b4cfc41320	- 61aea57497ad59b4cfc413ca
	- 61aea55c97ad59b4cfc4128c	- 61aea57397ad59b4cfc413a5
	- 61aea55f97ad59b4cfc41306	- 61aea57097ad59b4cfc4135e
	- 61aea55e97ad59b4cfc412ed	- 61aea57497ad59b4cfc413f2
	- 61aea55c97ad59b4cfc412a9	- 61aea57497ad59b4cfc413dd
DEV	granted:	denied:
	- 61aea55f97ad59b4cfc41332	- 61aea57497ad59b4cfc413e3
	- 61aea55d97ad59b4cfc412cc	- 61aea57497ad59b4cfc413b7
	- 61aea55d97ad59b4cfc412cf	- 61aea57197ad59b4cfc41379
	- 61aea55e97ad59b4cfc412f2	- 61aea57097ad59b4cfc41360
	- 61aea55c97ad59b4cfc41297	- 61aea57197ad59b4cfc41371
	- 61aea55e97ad59b4cfc412d6	- 61aea57497ad59b4cfc413c7
	- 61aea55e97ad59b4cfc412e2	- 61aea57497ad59b4cfc413bb

Sentence Segmentation

Step 2.1 - standard segmentation analysis:
We used the spacy sentence segmenter ("sentencizer") in english ("en_core_web_sm").
The results are the following:
mean precision = 0.05260126639321707 / standard deviation: 0.020939703572884355

mean recall = 0.05294471964676263 / standard deviation: 0.020358695419650005
mean f1 = 0.05261916553607473 / standard deviation: 0.020323027672159693
documents with min precision:
- 0.015748031496062992 from 61aea55f97ad59b4cfc41323
- 0.017391304347826087 from 61aea57497ad59b4cfc413ba

documents with min recall:
- 0.014925373134328358 from 61aea55d97ad59b4cfc412cb
- 0.015037593984962405 from 61aea55f97ad59b4cfc41323

We observe errors on:
- spatial markers such as \n \r \t or combinaison of them, especially when there is no white spaces between them and the other tokens
- footnotes '_____' are separated as '_', '_', '_', '_', '_', '_', '_', '_', '_'
- titles that are gathered with the following sentences

Step 2.2 - improved segmentation analysis:

For the error analysis, I selected the 2 documents with the lowest precision and the 2 documents with the lowest recall. It appears that the document with the lowest precision is also the second document with the lowest recall. Consequently, in the table we report only the numbers for the 3 documents.

We added custom sentence segmentation rule "\r\n\r", "_" for footnote

	61aea55f97ad59b4cfc41323	61aea57497ad59b4cfc413ba	61aea55d97ad59b4cfc412cb	All corpus
standard	p: 0.01574 r: 0.01503 f1: 0.01538	p: 0.01739 r: 0.01724 f1: 0.01731	p: 0.01785 r: 0.014925 f1: 0.01626	p: 0.05260 r: 0.05294 f1: 0.05261
with spatial separator handling	p: 0.12727 r: 0.21052 f1: 0.15864	p: 0.08290 r: 0.13793 f1: 0.10355	p: 0.10687 r: 0.20895 f1: 0.14141	p-- 0.108(μ), 0.039(σ) r--0.201(μ), 0.105(σ) f1-- 0.139(μ), 0.056(σ)
with footnote handling	p: 0.07777 r: 0.10526 f1: 0.08945	p: 0.03973 r: 0.05172 f1: 0.04494	p: 0.15217 r: 0.20895 f1: 0.17610	p– 0.107(μ), 0.039(σ) r-- 0.153(μ), 0.063(σ) f1-- 0.126(μ), 0.048(σ)

Footnote handling lowers the accuracy. We explain that as without this footnote handling, at the end of the documents, sentences' start are every integer. Consequently, some are matching the true sentences' start. But that doesn't mean that the sentence segmentation is worse, in fact it's better in the sense that all '_' are now together. Overall, our improved segmenter performs significantly better: from 0,015 up to 0,17 f1-score which is 10 times better. However, this is still not sufficient.

Step 2.3 - law-specific sentence segmenter:

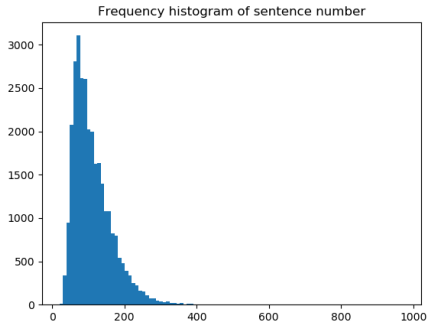
Using the law specific segmenter [1], the results are much better:
PRECISION--- mean: 0.836346060726036 standard deviation: 0.10285446268974077
RECALL--- mean: 0.9903319148502591 standard deviation: 0.012540009150727902
F1--- mean: 0.9031919280911297 standard deviation: 0.06806701274500085
We observe errors on:
- spatial markers such as \n \r \t or combinaison of them again. For example, "[...] Regional Office in \r\nPhiladelphia [...]" is segmented as "[...] Regional Office in." and "Philadelphia [...]"
- words in capital letters are considered as sentences, this is relevant for titles but not for tokens such as "ATTORNEY FOR THE BOARD"
- some citations are over segmented because of the presence of points. For example, "38 U.S.C.A. ﷺ 1110, 1131; 38 C.F.R." separated from "﷽ 3.303(a)."

We note that the worst precision score is 0,36 while the worst recall is 0,91. This highlights that the segmenter suffers mainly from over-segmentation (unmatched generated splits are counted as false positive which is at the denominator of the precision score). This is relevant regarding our error examples that spot over-segmented sentences.

Step 2.4:

A simple error analysis, highly suggest to use the Savelka's law-specific sentence segmenter (mean F1 score of 0.903 compared to the 0.108 for enhanced standard sentence segmenter).

Preprocessing



Step 3.1 - splitting unlabeled data:

The total number of sentences in the unlabeled corpus, after segmentation, is 3360495 ~ 3M. We present here the histogram corresponding to the number of sentences per document. The histogram's bin 70-80 is the one with the highest number of samples. Most of the documents have between 30 and 300 sentences, this is coherent with the observed documents during the annotation phase.

Step 3.2 - sentence wise preprocessing:

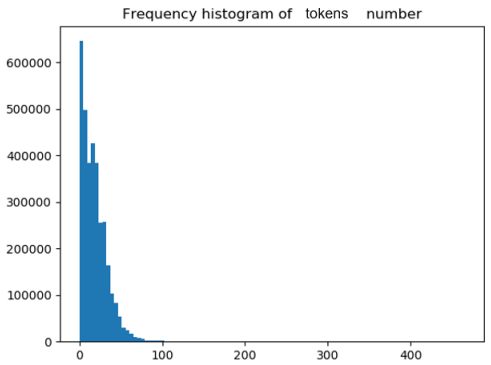
The baseline function used for this step is the spacy_tokenise function of the workshop. This function already handles punctuation (removal) and numbers (replaced by <NUMx> with x number length). To this we added:

- lowercase of everything
- possessiv 's removal (ex: veteran's claim was ['veteran', ''s', 'claim'] and is now ['veteran', 'claim'])
- handling of parenthesis in references (ex: 5103(a was seen as a <NUM> and tokenized as <NUM6> while 3.159(b was not seen as <NUM> and then was tokenized as '3.159(b'. Now we also have <NUM6> for 3.159(b). This gives more homogeneity to the tokenization
- removing punctuation characters (' , \n , \r , \t , [,]) with ruler of nlp
- removing the \n character of the tokens (ex: \n\n token was kept, we removed it. \ntoday token becomes today).
- handling of some common expression, tokens are kept together as one (ex: Veterans Law Judge, Vet. App., Veterans' Appeal, Veterans Affair)

These new rules were built based on the analysis of some example sentences (coming from the workshop notebook and from the segmented document from the previous step). The analysis code is gathered in "tokenisation_analysis.ipynb".

Step 3.3 - tokenize unlabeled data

Based on the tokenizer implemented in the previous step, we tokenized the unlabeled corpus in 3 steps:



1) Tokenization:
    a) creating a json file with for each doc a dictionary where each key correspond to a sentence
    b) creating a list containing number of tokens for each sentence
2) Plotting the histogram
3) Randomly ordered tokenized sentences:
    a) reading the json file created in 1)
    b) randomly selecting a document key (random.choice)
    c) randomly selecting a sentence in this selected document (random.choice)

d) write on a new line of tokenized_sentences.txt, the sentence only if it has at least 5 tokens (each token is separated by a single white space)
e) deleting this sentence key
f) deleting this document key if it was the last sentence of the document

We present here the histogram corresponding to the number of tokens per sentence. The histogram's bin with the highest number of tokens is < 5 tokens. We think that this is due to the fact that sentences at the beginning (for instance "Citation Nr: 1127293", "Decision Date: 07/21/11", "Archive Date: 07/29/11", "DOCKET NO. 07-22 055", "DATE", and titles) are common to all documents. If we consider that each document has 5 titles sentences: (5_title + 5_begin)* 30000_doc = 300000 which is on the scale of the pic of sentences with length < 5. This pic, at first sight strange, is easily explained.

Custom Embeddings

Step 4.1:

We trained FastText on our corpus of tokenized sentences from our preprocessing. The number of words announced when training a fasttext model is 62M > 60M, which is relevant. For all models, the number of words was 12634, the vectors' dimension was 100 (dim parameter), the minimum word occurrence count was set to 20 (minCount parameter). The file corresponding to this step is word_embeddings.py. We tested 3 models on training and we saved them for using them in the classification part:

- cbow for 10 epochs (FT 1)
- skipgram for 10 epochs (FT 2)
- skipgram for 20 epochs (FT 3)

Step 4.2 - evaluating custom embeddings manually:

We present here a list of words, their 6 nearest neighbors (in decreasing order), and observations. They are observed of training of FT 2 if not otherwise stated:

- **veteran** [(0.69, 'appellant'), (0.68, 'his'), (0.66, 'the'), (0.64, 'he'), (0.61, 'that'), (0.60, 'furthermore')]: neighbors are words that appear close to the word veteran, veterans are often 'appellant' in our corpus
- **v.** [(0.90, 'vet.'), (0.90, 'app.'), (0.84, 'app'), (0.80, 'brown'), (0.78, 'f.3d'), (0.77, 'nicholson')]: strange token 'f.3d' and close 'app' and 'app.'. Other tokens are representative of citations
- **argues** [(0.69, 'argue'), (0.59, 'holbrook'), (0.56, 'appealed'), (0.56, 'accredited'), (0.56, 'agree'), (0.55, 'ihp')]: close verbs in form ('argue') and meaning ('accredited'), no clear connection with ihp
- **argues** from FT 1 [(0.76, 'argue'), (0.58, 'arguendo'), (0.46, 'ihp'), (0.45, 'argument'), (0.44, 'accredited'), (0.44, 'agree')]: neighbors closer to radical form 'argu' than observed after FT 2
- **ptsd** [(0.75, 'depressive'), (0.74, 'mdd'), (0.74, 'pstd'), (0.72, 'bipolar'), (0.71, 'mst'), (0.71, 'dysthymic')]: spelling error 'pstd', and kinds of ptsd ('bipolar', 'depressive', …) presented which appear logical
- **granted** [(0.67, 'unwarranted'), (0.62, 'granting'), (0.58, 'grant'), (0.55, 'connection'), (0.54, 'bifurcate'), (0.53, 'theory')]: neighbors with same meanings and strange neighbors such as 'bifurcate'
- **korea** [(0.81, 'korean'), (0.79, 'dmz'), (0.77, 'demilitarize'), (0.77, 'germany'), (0.76, 'demilitarized'), (0.74, 'station')]: presence of 'demilitarize' and 'dmz' surprising
- **korea** from FT 1 [(0.83, 'korean'), (0.78, 'vietnam'), (0.78, 'koreans'), (0.73, 'germany'), (0.71, 'overseas'), (0.71, 'iraq')]: neighbors are other places, this is more logical than for FT 2 as it captures the meaning of the word not only it surroundings
- **holding** [(0.65, 'supra'), (0.62, 'ruling'), (0.61, 'clemons'), (0.60, 'kent'), (0.60, 'dingess'), (0.57, 'mandate')]: no clear patterns among neighbors

- **holding** from FT 1 [(0.58, 'supra'), (0.54, 'ruling'), (0.54, 'court'), (0.53, 'hold'), (0.46, 'holder'), (0.45, 'nicholson')]: ‘hold’ and ‘holder’ neighbors close but not the closest of the list. Again presence of this first ‘supra’ neighbor, I can’t explain
- **also** [(0.56, '<NUM4>'), (0.56, 'that'), (0.55, 'see'), (0.55, 'prinicpi'), (0.54, 'addition'), (0.54, 'atd')]: neighbors are not so close, only 0,56 for the closest, other small tokens
- **claim** [(0.60, 'file'), (0.59, 'evidence'), (0.57, 'connection'), (0.57, 'must'), (0.56, 'the'), (0.52, 'adjudicate')]: neighbors words are often associated to ‘claim’ in sentences
- **claim** from FT 1 [(0.72, 'proclaim'), (0.67, 'evidence'), (0.64, 'claimed'), (0.63, 'connection'), (0.59, 'necessary'), (0.58, 'information')]: neighbors with the same ‘claim’ radical, more logical than the observations with FT 2
- **benefits** [(0.71, 'vbms'), (0.67, 'paperless'), (0.67, 'virtual'), (0.64, 'vva'), (0.61, 'electronically'), (0.59, 'electronic')]: no clear patterns can be found
- **benefits** from FT 1 [(0.71, 'benefit-'), (0.52, 'vbms'), (0.51, 'beneficiary'), (0.50, 'virtual'), (0.49, 'paperless'), (0.47, 'pro')]: ‘benefi’ radical (‘beneficiary’, ‘benefit-’) logical, presence of ‘-’ that suggests a tokenization error
- **disability** [(0.75, 'disability-'), (0.64, 'current'), (0.60, 'connection'), (0.59, 'connect'), (0.59, 'service-'), (0.58, 'injury')]: presence of the ‘-’ in 2 of the neighbors likely associated based on it, highlights again a tokenization error. ‘injury’ connection with same meaning

From the presented observations, we note that there are noticeable differences comparing nearest neighbors from FT 1 and FT 2 models. FT 1 seems to be more efficient as it associates words with close meanings and radical forms. FT 2 gathers words a lot on their proximity in sentences. This analysis suggests continuing working with FT 1.

Training & Optimizing Classifiers

Step 5.1 - TFIDF featurization:

Tf-idf featurization was implemented based on the code of the workshop. Our implementation is in file tf-idf_featurization.py. The splitting between train, test and dev set was done accordingly with the dataset splitting section above. The dimension of the tf-idf featurization is (14292, 3175).

We present here the top 11 tokens after TFIDF vectorization for the Citation class, for the LegalRule class, and for one example sentence:

Citation		LegalRule		Example	
feature	tfidf	feature	tfidf	feature	tfidf
38	0.211844	the	0.093069	highlighted	0.351837
vet	0.093054	or	0.084154	audiologist	0.342588
app	0.092991	service	0.072669	substantive	0.308063
see	0.091001	of	0.069757	doctor	0.290660
303	0.084592	evidence	0.068302	performed	0.275055
west	0.051223	is	0.054712	form	0.259625
159	0.038874	be	0.054028	va	0.254396
2002	0.038457	disease	0.053462	september	0.207378
1110	0.038308	to	0.051732	2005	0.206281
5107	0.037803	disability	0.050198	november	0.204912
brown	0.036342	medical	0.038942	2006	0.189474

For example, we note that numbers are highly representative of Citations (6/11 of the best) and that 38 coming from “38 C.F.R.” has a high score (0,21) compared to others (<0,09). In LegalRule, we note the presence of stop words (the, or, of, …) among the best tokens, there are a lot of them (5/11). These tokens can be considered to have low informational value, therefore the preprocessing step of stop word removal could be considered.

We now look at an example sentence “In his September 2006 substantive appeal (VA Form 9), the veteran highlighted that the November 2005 VA examination was performed by an audiologist and not a medical doctor.” Overall, the best tokens of TFIDF make sense, no major error is seen here: “highlighted”, “performed” and dates give valuable information for classification.

Step 5.2 - word embedding featurization:

From the trained models in step 4.1, giving vectors of dimension 100, for each sentence, we computed the average of the embedding vectors in the sentences and we added two additional dimensions:

- a single float variable with the [0-1] normalized position of the sentence in the document
- a single float variable representing the number of tokens in the sentence, normalized by subtracting the mean and dividing by the standard deviation across all sentence tokens counts in the training data

The final embedding vectors were therefore of dimension 102. This step is mainly implemented in the word_embedding_featurization.py file.

Step 5.3 - model training:

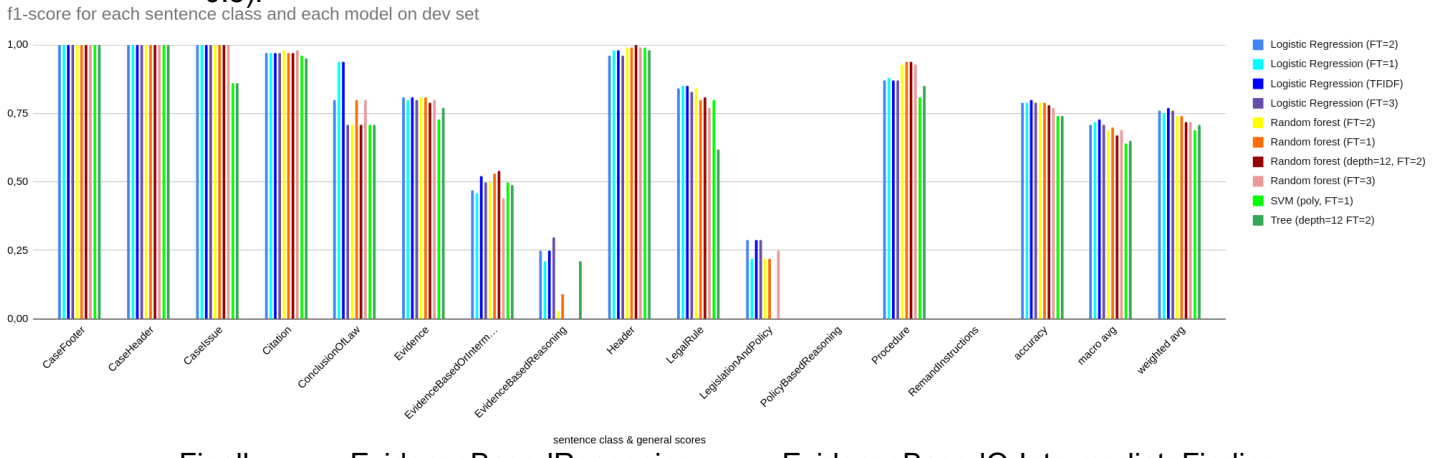
The choice of our best model was taken after the 3 following sets of experiments:

1) Training different kinds of models linear and nonlinear

As for linear models, we trained Logistic regression models with different word embeddings (fasttext models 1 & 2, and TF-IDF) settings. It appears that when looking at f1-scores, Logistic Regression model is one of the best models. For nonlinear models, we trained a polynomial SVM, a decision tree with depth limited to 12, and several random forest classifiers (fasttext models 1 & 2, and tree depth limited to 12). The results for the best models of their category are shown in the following graph, the scores reported are from tests of the dev set. For each sentence class, it presents the f1 score, and it also presents the global accuracy, macro average and weighted average for each model. We reported here only f1-score for clarity. We choose f1-score because it is made from both precision and recall scores. Of course, during our analysis we also took into account recall and precision.

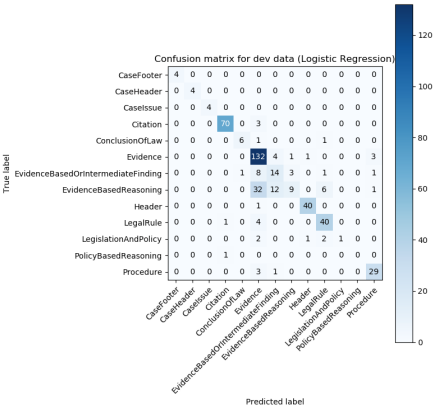
Based on this graph, we see that f1-scores highly depend on classes.

- CaseFooter, CaseHeader, CaseIssue, Citation, Header classes have almost perfect scores (f1-scores > 0.95 for all models). Considering only these classes don’t show clear difference between models and settings, we therefore didn’t focus on them for model improvements.
- ConclusionOfLaw, Evidence, LegalRule, Procedure classes have satisfying f1-scores (almost >0.70 for all models). However, for these classes a lot more differences appear across models. For instance, when considering ConclusionOfLaw, Logistic regression performs significantly better (f1-score = 0.94) than other models (f1-score < 0.8).



- Finally, EvidenceBasedReasoning, EvidenceBasedOrIntermediateFinding, LegislationAndPolicy, PolicyBasedReasoning and RemandInstructions classes have poor f1-scores (almost < 0.5 for all models). However, there are two different reasons

for poor performance. For EvidenceBasedReasoning, EvidenceBasedOrIntermediateFinding, the reason for poor performance is the fact that these two classes are close and then confusion appears between one another and also with the Evidence class. We also see the confusion between the three classes in the confusion matrix. For example, among the 60 EvidenceBasedReasoning sentences, 32 are classified as Evidence but only 9 are correctly classified. For LegislationAndPolicy, PolicyBasedReasoning and RemandInstructions classes, the reason for poor performance is lack of data (respectively 160, 27 and 3 in the training set and 6, 1 and 0 in the dev set). In addition, as experienced during the annotation part, these 5 classes were more difficult to annotate than classes such as Header.



It is also interesting to note that random forest classifiers perform significantly better than all other architecture for Procedure class.

In this part, we conducted training with the 3 different fasttext models (FT 1, 2, 3). Based on the graph, we selected FT 1 (significantly better results for ConclusionOfLaw with Logistic Regression and Random Forest).

2) Selection of the best non-linear model and tuning of hyper-parameters via a grid search approach

As a nonlinear model, we selected the Random Forest Classifier, as this model had the best results in our preliminary experiments. We processed with a grid search approach, training a classifier for each combination of parameters. First, we took the following parameters:

```
param_grid = {'n_estimators': [10, 40, 70, 100, 130],'criterion': ['gini', 'entropy'],
'max_depth': [12, 20, 30, 40]}
```

Then, as the best dev scores were obtained with 40 estimators and 30 as a max tree depth, we did a second grid search with the following parameters:

```
param_grid = {'n_estimators': [35, 40, 45],'criterion': ['gini', 'entropy'],'max_depth':
[25, 30, 35]}
```

From these experiments we can note:

- We observe overfitting for large numbers of estimators and large depth values (i.e. training scores to 1 but not so good evaluation scores). This is a common problem of models based on Decision Tree.
- The scores (on dev set) are not constant for the same model from one training to the other, and the differences make it difficult to choose the model. This is likely due to the small dataset we are using (especially, some classes have few samples). Cross validation is appropriate in that case, and this is why we use it in the next step.

The three models we selected are with parameters [45, gini, 30], [40, entropy, 30], and [40, gini, 30].

3) Selection of the 3 best sets of parameters and cross validation on them

With the three best sets of classifiers we conducted a 10-folds cross-validation approach. We found that the results are equivalent for each classifier with accuracy of 0.82 (mean) and 0.01 (std) on training and 0.78 on dev set. We selected the one that is the least complex and that has slightly better results: [n_estimators = 40, entropy, max_depth = 30].

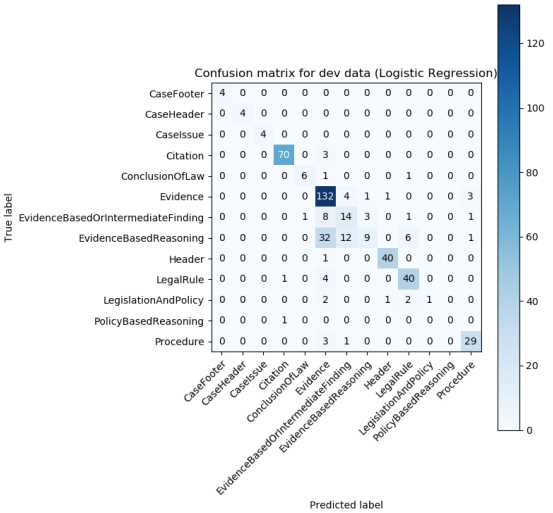
We note that the simple and linear Logistic Regression model performs as good as the best random forest classifier selected. Therefore, I would prefer the Logistic Regression model for its reduced complexity. We compared the running time of both models for training and prediction: the random forest classifier (8.54 sec) is 4 times longer than the logistic regression classifier (2.17 sec). This is a quantitative consideration that also favors the Logistic Regression algorithm.

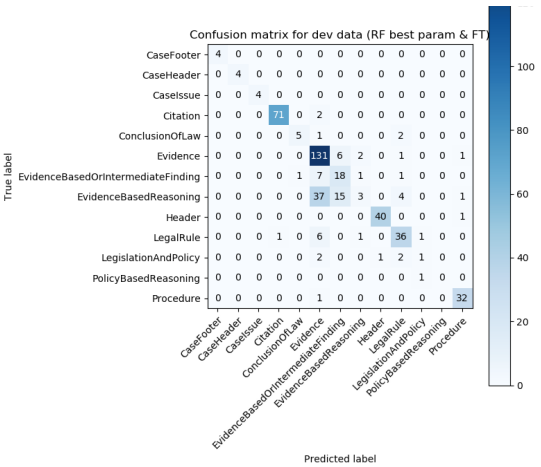
For the TF-IDF featurization, we used the same approach. However, for sake of concision we don't detail the process. The best model with TF-IDF featurization is Logistic Regression (better performance than Random Forest).

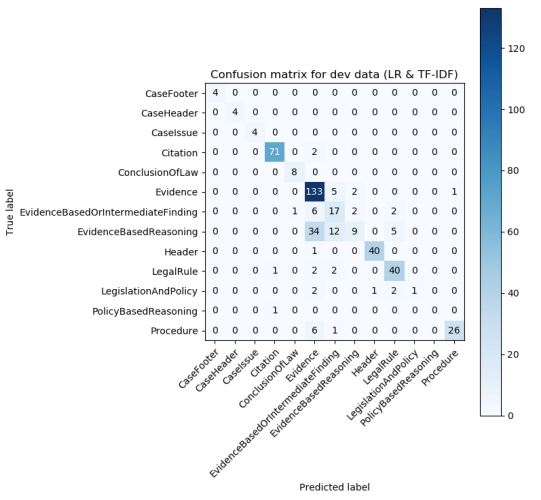
Test Set Evaluation

Model	Featurization	accuracy	macro average	weighted average
Logistic Regression	Fasttext	0,77	0,67	0,74
Random Forest [40, entropy, 30]	Fasttext	0,76	0,67	0,74
Logistic Regression	TF-IDF	**0,78**	**0,68**	**0,76**

First, the table shows that the scores on the testing set are comparable to the one we observed in the previous section for the development set. This is positive regarding the quality of our model selection steps.
Second, the best model is the Logistic Regression model with TF-IDF featurization. As what we observed in our literature survey, outside of deep learning, simple models tend to perform better than more complex models for this legal multi-class classification task. As explained in the Code instruction section, this is the default model used by our analyze.py file. However, the two other models can also be tested with special command line arguments.
Finally, we note that the weighted average (~0,75) is much higher than the macro average (~0,67) in all models. In fact, the weighted average weight class specific scores by the representation of the class in the dataset. Therefore, under-represented classes that are also the ones that have poor results (see 5.3) count less in weighted average than in macro average. This observation is again easily explainable. We present here the confusion matrix for the 3 models. Observations from section 5.3 are valid for them too.







Error Analysis

As said in 5.3.2 and as we see in the confusion matrix, there are lots of misclassifications between Evidence, EvidenceBasedOrIntermediateFinding, and EvidenceBasedReasoning classes. There were also confusions between these three classes during the annotation step. We selected false positive classification examples for each of these classes:

- Predicted Evidence (true label: EvidenceBasedReasoning): `"At the time of discharge, his psychiatric \r\nevaluation was normal."` I also would have annotated this sentence as Evidence. Therefore, I consider that the error comes from a human annotation error.

- Predicted EvidenceBasedReasoning (true label: EvidenceBasedOrIntermediateFinding): "Based on the evidence verified by the service department \r\nrecords, service medical records, and the historical record, \r\nand given the current diagnosis of PTSD, which has been \r\nmedically linked to active duty service, with a stressor \r\nconsistent with service, the Board finds that the evidence is \r\nsufficient to corroborate that the stressor actually \r\noccurred." The last part of the sentence with "the Board finds" shows that this is an intermediate finding (like the true label). The reason for misclassification may be that this is a long sentence with the first part representing the reasoning steps behind the finding, therefore the words of the 1st part may suggest reasoning type.
- Predicted EvidenceBasedOrIntermediateFinding (True label: EvidenceBasedReasoning): "This evidence establishes sufficient \r\nverification of his alleged personal exposure to stressful \r\nevents during military service." I also would have annotated this sentence as EvidenceBasedOrIntermediateFinding as the verb "establishes" suggests. Therefore, I consider that the error comes from a human annotation error.

Some errors occurs outside of these 3 classes, and here are some examples:

- Predicted LegalRule (True label: LegislationAndPolicy): "The Veterans Claims Assistance Act (VCAA)" This sentence is not complete, therefore the error is likely to come from sentence segmentation error.
- Predicted Citation (True label: LegalRule): "Upon receipt of a complete or \r\nsubstantially complete application, VA must inform the \r\nclaimant of information and medical or lay evidence not of \r\nrecord: (1) necessary to substantiate the claim; (2) that VA \r\nwill seek to obtain; (3) that the claimant is expected to \r\nprovide; and (4) request the claimant to submit any relevant \r\nevidence in the claimant's possession in accordance with 38 \r\nC.F.R. • 3.159(b)(1). 38 U.S.C.A. • 5103(a) (West 2002 & \r\nSupp. 2005); 38 C.F.R. • 3.159(b) (2006); Charles v. \r\nPrincipi, 16 Vet. App. 370, 373-74(2002); Quartuccio v. \r\nPrincipi, 16 Vet. App. 183, 186-87 (2002)." This sentence is truly a LegalRule sentence but also contains a citation at the end, the markers for the citation sentence are likely to have more weight than other words from the rest of the sentence. In fact, "38 C.F.R" is common in citation sentences, and as we saw in 5.1, '38' is highly representative of citations. In addition, the following sentence, which is a citation, is also part of the sentence. This misclassification is linked to two elements: error of the sentence segmentation and citation tokens at the end of the sentence.
- Predicted LegalRule (True label: ConclusionOfLaw): "PTSD was incurred as a result of active service. 38 U.S.C.A." This sentence is likely to have been misclassified because of three reasons: "38 U.S.C.A" tokens from a sentence segmenter error, "PTSD" token that often appear in LegalRule (other examples of misclassified sentence as LegalRule contain the "PTSD" token), and whereas a lot of ConclusionOfLaw sentences there is no "granted" or "denied" token.

Discussion

Main findings summarization:

- Importance of the quality of the dataset (annotations), of the number of the samples in each class (models perform poorly on under-represented classes)
- Law specific segmenter has better results by far but some sentence segmentation errors remain (mainly over-segmentation) and cause misclassification at the end.

- Spacy based tokenizer gives promising results, but at this stage fails to handle completely spatial separators (\n\r\t), especially when there is no whitespace between them and actual sentences.
- TF-IDF and FastText embeddings have comparable results
- Simpler models (Logistic Regression) perform better on our dataset than nonlinear machine learning models (polynomial SVM, and Random Forest to some extents)
- Some types are easier (CaseFooter, CaseHeader, CaseIssue, Citation, Header, Procedure) to classify than others (EvidenceBasedReasoning, EvidenceBasedOrIntermediateFinding, LegislationAndPolicy, PolicyBasedReasoning and RemandInstructions). **For the easiest classes, which give the general structure of the document, f1-score is often close to one in our basic approach, therefore automatic classification of them can be considered immediately.** Sentence classes that are part of the reasoning for decisions are the most difficult to classify for our ML models, while document structure parts are easier.
- Poor class performance is found in two main situations: subjective annotation type and lack of sample in this class

Based on the presented experiments and discussions, I would suggest several directions for the future:

- Annotations: create a corpus with high quality annotations as it directly influences the performance of the model at the end. I would recommend several legal domain experts (note that 3 annotators is often adopted in the literature) that annotate the same texts and then do a majority voting to have the final class for each sentence. Discussions between annotators could be considered when there are samples with high disagreement.
- Gather more samples from the following classes: RemandInstructions, PolicyBasedReasoning
- Additional pre-processing steps especially concerning tokenization
- Continue with simple models or going into deep learning models, especially I would recommend Bi-LSTM based architecture based on the literature survey
- Continue with (10 folds) cross-validation
- Continue with TF-IDF featurization or try law specific vectorization such as Law2Vec [9]

Code Instructions

Information can be found under the README.md file concerning requirements and dependencies as well as description of the folder structure. You can test the model with a BVA txt file. To do so, you must write in the command line:

```
python analyze.py <path/file.txt>
```

There are also 2 optional arguments:
- "write_to_csv" (-csv): boolean: when False the output is printed in the terminal, when *True* the output is written in a csv file ('output_predicited_type.csv' in folder 'outputs') but no more in the terminal. Its default value is *False*.
- "model" (-m): string: The model you want to use for prediction. Either 'LR&TFIDF' for Logistic Regression with TF-IDF embeddings (default) or 'LR' for Logistic Regression with Fasttext embeddings or 'RF' for Random Forest with Fasttext embeddings.

To use these optional arguments, you must write in the command line, for example:

```
python analyze.py <path/file.txt> -csv True -m 'LR'
```

The output has on each line a sentence and its predicted type.

References

[1] Savelka, Jaromir, Vern R. Walker, Matthias Grabmair and Kevin D. Ashley. "Sentence Boundary Detection in Adjudicatory Decisions in the United States." TAL 58.2 (2017)[https://github.com/jsavelka/luima_sbd]

[2] P. Bojanowski*, E. Grave*, A. Joulin, T. Mikolov. "Enriching Word Vectors with Subword Information" TACL 5 (2017)[https://github.com/facebookresearch/fastText]

[3] Walker, Vern R. et al. “Automatic Classification of Rhetorical Roles for Sentences: Comparing Rule-Based Scripts with Machine Learning.” *ASAIL@ICAIL* (2019).

[4] E. de Maat, K. Krabben, and R. Winkels. “Machine Learning versus Knowledge Based Classification of Legal Texts.” *JURIX (2010).*

[5] B. Hachey , and C. Grover “Sentence Classification Experiments for Legal Text Summarisation” *JURIX (*2004)

[6] Bhattacharya, P., Paul, S., Ghosh, K. *et al.* DeepRhole: deep learning for rhetorical role labeling of sentences in legal case documents. *Artif Intell Law* (2021)

[7] S. R. Ahmad, D. Harris and I. Sahibzada, "Understanding Legal Documents: Classification of Rhetorical Role of Sentences Using Deep Learning and Natural Language Processing," *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, (2020).

[8] Glaser, Ingo et al. “Classifying Semantic Types of Legal Sentences: Portability of Machine Learning Models.” *JURIX* (2018).

[9] I. Chalkidis, “Law2Vec: Legal Word Embeddings" [https://archive. Cross-lingual Annotation Projection in Legal Textsorg/details/Law2Vec]

[10] S. Undavia, A. Meyers and J. E. Ortega, "A Comparative Study of Classifying Legal Documents with Neural Networks," *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2018

[11] Nazarenko, Adeline & Lévy, François & Wyner, Adam. (2021). A Pragmatic Approach to Semantic Annotation for Search of Legal Texts – An Experiment on GDPR. 10.3233/FAIA210313.

[12] Claire Grover, Ben Hachey, Ian Hughson, and Chris Korycinski. 2003. Automatic summarisation of legal documents. In *Proceedings of the 9th international conference on Artificial intelligence and law* (*ICAIL '03*). Association for Computing Machinery, New York, NY, USA, 243–251.

[13] Dragoni, Mauro & Villata, Serena & Rizzi, Williams & Governatori, Guido. (2016). Combining NLP Approaches for Rule Extraction from Legal Documents.

[14] Andrea Galassi, Kasper Drazewski, Marco Lippi, and Paolo Torroni. 2020. Cross-lingual Annotation Projection in Legal Texts. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 915–926, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[15] Chalkidis, I., Kampas, D. Deep learning in law: early adaptation and legal word embeddings trained on large corpora. *Artif Intell Law* **27,** 171–198 (2019).

[16] Ilias Chalkidis, Emmanouil Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-Scale Multi-Label Text Classification on EU Legislation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6314–6322, Florence, Italy. Association for Computational Linguistics.

[17] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

[18] Ilias Chalkidis, Manos Fergadiotis, and Ion Androutsopoulos. 2021. MultiEURLEX - A multi-lingual and multi-label legal document classification dataset for zero-shot cross-lingual transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6974–6996, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[19] Guido Boella, Luigi Di Caro, Valentina Leone. Semi-Automatic Knowledge Population in a Legal Document Management System, 2019.

[20] Robaldo, L., Villata, S., Wyner, A. *et al.* Introduction for artificial intelligence and law: special issue “natural language processing for legal texts”. *Artif Intell Law* **27,** 113–115 (2019).

[21] A. Wyner. Towards Annotating and Extracting Textual Legal Case Elements, 2010