Journal of Machine Learning Research 1 (2020) 1-48 Submitted 4/00; Published 10/00

,

Text classification for the juridical field

Saad TAZROUTE Saad.tazroute@centrale-casablanca.ma
Ecole Centrale Casablanca
Casablanca, Morocco

Editor:

Abstract

In this article, we examine the application of text classification approaches to assist legal professionals. We present several experiments applying classification techniques to predict the French Court of Cassation ruling and the area of law to which a case belongs.

We work on the CASS data set, composed of the judgments of the French Court of Cassation, containing the body of case law, and the ruling decision and others informations.

We use two methods for classification. The first one is based on the TF-IDF score for feature extraction from the law text, we used these features as entries to several machine learning algorithms. The second method consists of using a pre-trained model to extract features named BERT based on Transformers architecture, and specially its French version named CamemBERT. We used especially in this paper the stacking of this architecture and a classification layer to realise the classification task.

Due to resources limitations, we trained the CamemBERT model only on 10 000 case law.

We report results of 84% F-1 score in predicting a case ruling using CamemBERTforSequenceClassification , 89% F-1 score using the Tf-IDF score and a linear Support Vector Machine (SVM) classifier trained on lexical features.

Keywords: Multi-class Text Classification, Term frequency/inverse document frequency (TF-IDF), Bag of words,BERT , Transformers Architecture , CamemBERT.

1. Introduction

Text classification methods have been successfully used in a number of NLP tasks and applications ranging from plagiarism detection [**Barrón-Cedeño et al. (2013)**] to imitation identification [**Dinu et al. (2012)**].

It was also used in order to enhance the efficiency of recommendation system through user profiling [**Roozmand et al. (2011)**] and sentiment analysis, identifying potential criminals [**Sumner et al. (2012)**], crimes [**Pérez-Rosas and Mihalcea (2015)**], or anti-social behavior [**Cheng et al. (2015)**].

In this paper, we focus on classifying legal text. The contribution of IA in the legal field nowadays becomes more important, and the legal field represents an emerging NLP field with many applications (e.g., legal judgment [**Nallapati and Manning (2008)**], contract element extraction [**Chalkidis and Androutsopoulos (2017)**], obligation extraction but limited publicly available resources.

TF IDF scoring was used in several applications related to text mining, [**Terachi et al. (2006)**], and represents the backbone of feature extraction to feed Machine learning algorithms.

At the end of 2018, the artificial intelligence research community has made significant progress the development of NLP techniques based on deep learning. This is due to the publication of the article "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by the Google AI team [**Devlin et al. (2018)**].

BERT is based on the transformer architecture [**Vaswani et al. (2017)**], which is mainly based on the attention mechanism and defines a new type of deep neural network arrangement. Monitored by the open sourcing of the Transformers architecture by tech firms like Huggingface [**Wolf et al. (2019)**] , academics working with NLP in-depth learning methods[**Stickland and Murray (2019)**] have been able to replicate such results, while refining the model for their own research tasks.

Many disciplines generate an extremely rich volume of natural language data, although the social sciences and humanities maintaining the greatest importance. Among the areas dealing with human culture and society, the area of law and politics remains the most complex to manage, since the human factor remains present throughout the process of value creation and allows for some daily speculation questions about the basis of human actions. They represent a great challenge and are therefore good choices as areas of knowledge extraction. To highlight this aspect, and to be able to help structure these ideas, and to bring in techniques to reduce tedious tasks and to bring light and structure to otherwise very verbose and controversial data is one of the main objectives of the NLP.

In our work, we provide a multi-class text classification based on the case description. In order to do so, we acquire a large body of French judicial decisions with more than 130,000 documents from the 1800s to June 2020. We are exploring the use of two approaches to solving the problem. The first approach extracts features based on term occurrence (TF-IDF), and feeds different Machine learning algorithms. The second approach is based on a context-sensitive pre-trained model on a long corpus of the French language and we stack on the top of the network a softmax layer responsible for classification.

2. Data :

In this paper we represent the french CASS Data set, which gathers a collection of court rulings from the French Supreme Court (*La cour de cassation*). The complete collection available here [1] consists of **130.490** documents, each document consists of a unique court and describes the case in meticulous details.

2.1 Data description

The provided Data set describes each case by: a unique id of the case, the date of the decision, case ruling (*rejection*, *cassation*, *qpc* ..), a case description, and cited laws.

The data provided by the french CASS data set is a collection of XML structured files, the first step of preprocessing.

During pre-processing, we removed all duplicate and incomplete entries in the Data set. This resulted in a corpus consisting of **127.587** unique court rulings.

1. collected from `https://www.legifrance.gouv.fr`

Each instance contains a case description and four different types of labels: a law area, the date of ruling, the case ruling itself, and a list of articles and laws cited within the description

In this study, the text of each document has been cleaned by removing all HTML enclosing tags and trimming the trailing spaces and tabs. The resulting data is a CSV file with **127.587** case laws which is equivalent to **127.587** rows in our Data set preserving every field with the ability to process the cased corpus.

For classification task, ruling prediction, we carry out two sets of experiments. The experiment (6-class setup) considers only the first word within each label and only those labels which appeared more than 200 times in the corpus. This lead to an initial set of 6 unique labels: cassation, annulation, irrecevabilite, rejet, non-lieu, and qpc (question prioritaire de constitutionnalité).

Ruling type	**Number of cases**
Cassation	54665
irrecevabilité	2534
non-lieu statuer	139
rejet	69709
annulation	193
qpc	278

Table 1: Distribution of cases according to ruling type

We can see that the data set is extremely unbalanced. We take this in consideration in the evaluation metricsn, by taking the micro average of the F1 score, the weights will be based on the number of cases for each label.

2.2 Dealing with long texts :

An other problem that arises during the implementation phase is the length of the text.

We describe the number of words of each case description among all the 127 587 case descriptions as the following :

mean	989.95
std	1504.19
min	11
25%	441
50%	630
75%	986
max	104835

Table 2: Number of words per case description

To overcome this problem, we refer to the paper [**Sun et al. (2019)**], which provides 3 approaches to truncate a long text in order to use for BERT.

The maximum sequence length of BERT is 512. The first problem of applying BERT to text classification is how processing the text with a length larger than 512. We try the following ways for dealing with long descriptions.

Truncation methods

Usually, the key information of an article is at the beginning and end. We use three different methods of truncate text to perform BERT fine-tuning.

- **head-only:** keep the first 510 tokens
- **tail-only:** keep the last 510 tokens
- **head+tail: empirically select the first 128 and the last 382 tokens**

In this paper, it was based on the nature of the legal text, and the case description usually remains after the first 200 words in the text, and generally includes 350 words, so we focused on this rule based on the nature of the data set.

3. Problem formulation : Multi-class Text Classification

Let's consider a data set D containing sequences of text samples :

$$D = D_1, D_2, \ldots, D_N$$

We consider also a set of labels denoted as:

$$Y = Y_1, Y_2, \ldots, Y_M$$

- Where D_i is the i^{th} sample of our data set, which can be a document, a text segment.
- Y is a set of unique labels and M is the number of labels in the data set.

Classification is the task of approximating a mapping function (f) from input variables (D) to discrete output variables (Y):

$$f(D) = Y$$

In this paper, we consider that we want to classify the different documents of our data set, in other terms each row is a single document.

Each document has a unique class to which it belongs.

4. Methodology

4.1 First approach :

4.1.1 Bag of words :

Bag of words is a representation of the entire text corpus in feature sets, where the attributes are possible terms and the values are the occurrence of a word in the given document. It is the most used method for defining the most frequent words in a corpus, it helps to encode

the text as a term-document matrix, and thereafter this matrix gives the keys words of each category. **[Ren and Malik (2003)]**

Before starting this step, the pre-processing step is essential to reduce the size of our embedded text. The pre-processing step consists of deleting stop-words, lemmatization, stemming.

4.1.2 Pre-processing

In order to reduce the size of this matrix we pre-process our text in order to keep just the unique words.

- Removing stop-words.

  It's common in the pre-processing step to remove the empty words, i.e. a list of the most frequent words in a language that provides structure rather than of the content. Examples of these words include, prepositions and conjunctions. They are so common that they do not need to be indexed. In French, empty words evident could be " le ", " la ", " de ", " du ", " ce ". This technique reduces the size of the dictionary and thus the dimensionality of the representation of comments.

- Lemmatization

  Lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form.

- Stemming Stemming allows us to obtain a truncated form of the word, common to all morphological variants.

4.1.3 TF-IDF weight :

The first model we tested is the bag of words model with term frequency-inverse document frequency scores. Each row of our new data set represents a unique document. The documents are represented as vectors but instead of a vector of '0's and '1's which is a another approach called " one hot encoding" which is out of the scope of this paper. The second method of embedding which is the operation of transformation of a text into a vector. In this paper, we focus on this second method based on the term frequency- inverse document frequency (TF-IDF) score.

We obtain a matrix with a size $(N \times M)$ which M is the number of documents which we want to classify, N is the number of unique words in all the texts.

Now the document contains scores for each of the words. These score are calculated by multiplying TF and IDF for specific words.

Typically, the TF-IDF weight is composed by two terms: the first computes the normalized Term Frequency (TF). In this paper TF represents the number of times a word appears in a case description, divided by the total number of words in that case description; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- *TF*: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$tf_{i,j} = \frac{|\{t_i : t_i \in d_j\}|}{|d_j\|}$$

  - $|\{t_i : t_i \in d_j\}|$: Number of times term t_i appears in a document d_j
  - $|d_j\|$: Total number of terms in the document d_j

- *IDF*: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

  Where :

  - $|D|$: represents the total number of documents.
  - $|\{d_j : t_i \in d_j\}|$: number of documents where the term t_i appears

we obtain finally a weighted matrix which we can use it in classification algorithms who have proven their robustness in the literature.

we express the weight of each term i in a document j by :

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \cdot \mathrm{idf_i}$$

4.2 Second approach : Deep Learning pre-trained models : BERT

4.2.1 BERT : Bidirectional Encoder Representations from Transformers

4.3 BERT architecture :

BERT creates a new linguistic model for the English language, which takes into account the context in a sequence of words.

BERT makes use of **Transformer**, an attention mechanism that learns contextual relations between words (or sub-words) in a text.

In its original version, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task, see an example in the figure **[1]**.

Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google. **[Vaswani et al. (2017)]**

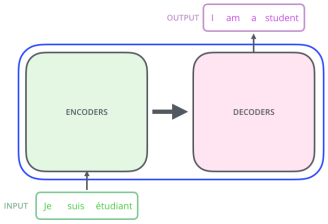

Figure 1: Example of transformer architecture for translation task[2]

The encoding component is an overlay of encoders. The original version of the paper stacks six encoders on top of each other.

The encoders are all identical in structure . Each one is broken down into two sub-layers, as shown in the figure **[2]**

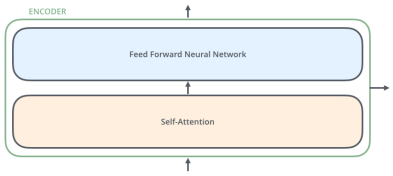

Figure 2: Encoder architecture taken from [3]

The encoder inputs first go through a self-attention layer - a layer that helps the encoder look at other words in the input sentence when encoding a specific word.

The outputs of the self-attention layer are transmitted to a direct-acting neural network. Here, the encoder maps an input sequence of symbol representations $X = (x_1, ..., x_n)$ to a sequence of continuous representations $Z = (z_1, ..., z_n)$.

Attention can be seen simply as a function that takes a sequence X as input and returns another sequence Z of the same length, composed of vectors of the same length of those in X.

$$Z = Attention(X)$$

A more precise explanation of the attention layer is well detailed in **Appendix A** .

4.3.1 CamemBERT : A french model of BERT

CamemBERT is a french version of BERT based on Facebook's RoBERTa model released in 2019[**Liu et al. (2019)**].

CamemBERT creates a new language model for French. Similar to RoBERTa [**Liu et al. (2019)**] (for English) and BERT [**Devlin et al. (2018)**], CamemBERT is a multi-layer bidirectional Transformer [**Vaswani et al. (2017)**]

In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A.

Each transformer block contains only an encoder like what we have in the BERT Architecture, each encoder is similar to figure **[2]**.

The main difference between the different models is the data on which the model is trained. CamemBERT is pretrained trained on 138 GB of French text, more specifically on the French sub-corpus of the newly available multilingual corpus OSCAR , which is a french monolingual corpus pre sented in **[Suárez et al. (2019)]**.

CamemBERT uses the original architectures of $BERT_{BASE}$($L = 12$, $H = 768$, $A = 12$, Total Parameters=110M) and $BERT_{LARGE}$ ($L = 24$, $H = 1024$, $A = 16$, Total Parameters=340M).

5. Implementation and results :

5.1 Implementation

5.1.1 Tf-idf score:

A sequence of text cannot be fed directly to the algorithms themselves as most of them expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.

CountVectorizer in the scikit-learn library implements tokenization and occurrence counting.**[Pedregosa et al. (2011)]**

In order to re-weight the count extracted by *CountVectorizer* into floating point values suitable for usage by a classifier, we use the TF-IDF score.

The TF-IDF score can be expressed as the following :

$$tf - idf(t, d) = tf(t, d) \times idf(t)$$

Where t is a term, d is a document.

The resulting tf-idf vectors are then normalized by the Euclidean norm:

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{{v_1}^2 + {v_2}^2 + \ldots + {v_n}^2}}$$

TfidfVectorizer is a feature extraction from text method implemented in the scikit-learn library which converts a collection of raw documents to a matrix of TF-IDF features. **[Pedregosa et al. (2011)]**

After that we use different machine learning algorithms in order to classify these vectors.

We used for the classification task several machine learning methods which we can list here.

5.1.2 CamemBERT

CamemBERT generates a language model, in other terms the model takes in input a text sequence and returns a vector representing the text.

Transformers library provides general-purpose architectures (BERT, RoBERTa,...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages.

Transformers is provided by Hugging Face firm, with a main goal which is democratizing NLP for the research community. [**Wolf et al. (2019)**]

CamembertforSequenceClassification is a CamemBERT Model transformer with a sequence classification/regression head on top (a linear layer on top of the pooled output).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
roberta (TFRobertaMainLayer) multiple                  110621952
_________________________________________________________________
classifier (TFRobertaClassif multiple                  592130
=================================================================
Total params: 111,214,082
Trainable params: 111,214,082
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: CamemBERTforSequenceClassification architecture

5.2 Results :

We represent the results obtained by the two approaches: TF-IDF with machine learning algorithms, and CamemBERT and specially the class CamemBERTforSequenceClassification.

Due to resource limitations, we trained all the methods on random sample of 10,000 case descriptions of the data set.

	TF-IDF + Random forest	TF-IDF + SVM	TF-IDF + Logistic regression	TF-IDF + Naive Bayes	CamemBERT
Accuracy	0.7725	0.8624	0.8533	0.8031	0.8333
F1-score (Weighted average)	0.7448	0.8652	0.8254	0.7854	0.8333

Table 3: We used two metrics in order to evaluate the performance of the several methods used in the state of art, the two metrics are Well detailed in appendix B

6. Discussion :

We conducted along this article a comparison between several methods for juridical text classification.

Even if it remains a fairly classic extraction tool, the TF-IDF method represents an important method for feature extraction.

We combined this tool, with a diversified set of machine learning algorithms known by their performance in this task

The Support vector machine algorithm perform better any other algorithm.

The second approach, remains extremely important to discuss and evaluate since it is based on neural networks. CamemBERT had very satisfactory results, comparing them with the state of the art results in the classification task with extremely unbalanced data.

In deep learning we need a huge amount of data, and maybe if we feed this neural network with all the data set, it will perform better.

An other point to discuss, is the data used in this experimentation, and the method of truncating the text leads to many advantages like reducing the number of features and make us able to compute the code on the 1/10 of the data.

Acknowledgments

I would like to express my sincere gratitude to my advisors from Ecole Centrale Casablanca Mr. **Jean-Pierre LLORED**, Mr. **Anass BOUCHNITA** , and Mr. **Christophe DENIS** my advisor of the computer science laboratory of the Sorbonne University (LIP6), and Mrs. **Bénédicte BEVIERE BOYER** from university Paris 8 for the continuous support during this project, for their patience, for their generosity and advice, for following my work during all its steps and also for sharing with me all the technical and theoretical knowledge for carrying out my project.

My sincere thanks also go to the teaching staff at **Ecole Centrale Casablanca** and especially to Mr. **Llored** to allow me to get this very formative opportunity. Finally, I thank all those who have contributed directly or indirectly to the achievement and success of this work. I hope that this work is on the horizon of your expectations as well as at the level of the efforts provided.

Appendix A. Attention mechanism

In this appendix, we describe the attention mechanism from Section 4.2.1 :

The first step in calculating self-attention is to extract three vectors from each of the encoder's input vectors, which is the embedding of each word.

In a text sequence denoted as $X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{pmatrix}$ a matrix of size $(N \times M)$ where N is the length of the sequence and M is the embedding dimension (each word is represented by a vector of size M that we note X_i).

1. Extracting 3 vector from each word vector : For each word X_i, we create 3 vectors, a **Query** vector, a **Key** vector, and a **Value** vector which we denote respectively Q_i, K_i, V_i

$$Q_i = X_i \times W^Q$$
$$K_i = X_i \times W^K$$
$$V_i = X_i \times W^V$$

Where W^Q, W^K, W^V are three trainable weight matrices in the attention layer.

Multiplying X_i by the W^Q weight matrix produces Q_i, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

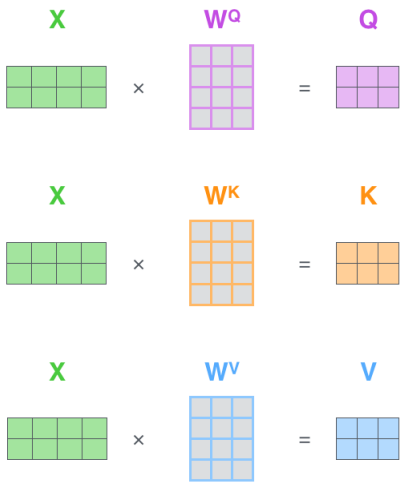

Figure 4: For each word X_i we obtain 3 vectors associated to the vector Q_i, K_i, V_i.

2. We calculate the score :

   BERT uses a *compatibility* function, which assigns a score to each pair of words indicating how strongly they should attend to one another.

   Like we said the output of the self-attention block is a weighted sum vector.

   The head computes attention weights w between all pairs of words as softmax-normalized dot products between the query and key vectors. The output Z of the attention head is a weighted sum of the value vectors.

$$w_{i,j} = Softmax(q_i^T k_j)$$

   Where the softmax function is described by : $Softmax(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

$$w_{i,j} = \frac{\exp(q_i^T k_j)}{\sum_{l=1}^{n} \exp(q_i^T k_l)}$$

   The idea behind multiplying each value vector by the softmax score (in preparation to sum them up). The intuition here is to keep intact the values of the word(s) we want to focus on, and give a negligible weight to irrelevant words (by multiplying them by small numbers like 0.001, for example).

3. The output vector :

$$Z_i = \sum_{j=1}^{n} w_{i,j} V_i$$

   Where V_i is the value vector of the i^{th} word in the word, and Z_i the i^{th} output of the self-attention layer.

   This process is done in parallel thanks to the multi-head attentions, which calculate the self attention in parallel for the whole sequence.

Appendix B. Metrics:

In order to evaluate the performance of each method, we used several metrics:

B.1 Confusion Matrix:

The confusion matrix is a handy presentation of the accuracy of a model with two or more classes.

The table presents predictions on the x-axis and accuracy outcomes on the y-axis. The cells of the table are the number of predictions made by a machine learning algorithm.

		True/Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Figure 5: The confusion matrix for a "One vs all" classification

B.2 Accuracy:

Classification accuracy is the number of correct predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems. In fact, it is only suitable when there are an equal number of observations in each class and that all predictions and prediction errors are equally important, which is often not the case. [4]

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN}$$

Appendix C. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. [5]

$$Precision = \frac{TP}{TP + FP}$$

Appendix D. Recall

Recall is the ratio of correctly predicted positive observations to the all observations in actual class .

$$Recall = \frac{TP}{TP + FN}$$

D.1 F1-score:

F-1 score is a measure of a test's accuracy. It considers both the precision p and the recall of the test. The F1 score is a better measure of a model's accuracy than just using accuracy especially when the model is imbalanced. To compute this metric:

$$F - 1score = \frac{Precision \times Recall}{Precision + Recall}$$

We had a look on the population of each label, the major remark that can be extracted is that the data is highly unbalanced.

A weighting is used to make an average of the F1-score based on the number of each label in the data set.

4. https://towardsdatascience.com/evaluation-metrics-for-classification-problems-in-machine-learning-d9f9c7313
5. https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

Remainder omitted in this sample. See http://www.jmlr.org/papers/ for full paper.

References

Alberto Barrón-Cedeño, Marta Vila, M Antònia Martí, and Paolo Rosso. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. *Computational Linguistics*, 39(4):917–947, 2013.

Ilias Chalkidis and Ion Androutsopoulos. A deep learning approach to contract element extraction. In *JURIX*, pages 155–164, 2017.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. Antisocial behavior in online discussion communities. In *Ninth International AAAI Conference on Web and Social Media*, 2015.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Liviu P Dinu, Vlad Niculae, and Octavia-Maria Şulea. Pastiche detection based on stopword rankings. exposing impersonators of a romanian writer. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 72–77, 2012.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ramesh Nallapati and Christopher D Manning. Legal docket classification: Where machine learning stumbles. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 438–446, 2008.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Verónica Pérez-Rosas and Rada Mihalcea. Experiments in open domain deception detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1120–1125, 2015.

X Ren and J Malik. the ninth ieee international conference on computer vision. *Nice, France: IEEE*, pages 10–17, 2003.

Omid Roozmand, Nasser Ghasem-Aghaee, Mohammad Ali Nematbakhsh, Ahmad Baraani, and Gert Jan Hofstede. Computational modeling of uncertainty avoidance in consumer behavior. *International Journal of Research and Reviews in Computer Science*, page 18, 2011.

Asa Cooper Stickland and Iain Murray. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. *arXiv preprint arXiv:1902.02671*, 2019.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. *Challenges in the Management of Large Corpora (CMLC-7) 2019*, page 9, 2019.

Chris Sumner, Alison Byers, Rachel Boochever, and Gregory J Park. Predicting dark triad personality traits from twitter usage and a linguistic analysis of tweets. In *2012 11th international conference on machine learning and applications*, volume 2, pages 386–393. IEEE, 2012.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

Masahiro Terachi, Ryosuke Saga, and Hiroshi Tsuji. Trends recognition in journal papers by text mining. In *2006 IEEE international conference on systems, man and cybernetics*, volume 6, pages 4784–4789. IEEE, 2006.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910, 2019.