Use of NLP Methodologies in predicting Vento bands

Usama Nadeem[1]

Abstract

The problem statement as it pertains to this work is as follows: how can we employ NLP methodologies to predict which Vento Band a particular case will fall into, from its case summary. Looking at it first as a similarity problem, we investigate on a small database of case summaries, the usual metrics of similarity coupled with Bag-of-Words and TD-IDF methodologies as a first step. We then follow this up with a more sophisticated tool of Doc2Vec account for more nuance and finally complete this programme by repeating the previous exercises with the vectorisation of words coming from Google's BERT.

In addition to the previous standard approaches, we also put forward popular learning methodologies, that include a Naive Bayes' Classifier, Support Vector Machines, Random Forest, and Multilayered Perceptron, finally concluding with recommending the state-of-the-art in NLP methodology for this task.

1. Introduction

In the nations of England and Wales, where a common-law based system is prevalent, the prescription of damages in legal cases is dependent on a variety of factors, mainly statute, guidelines, and precedent. For the cases, we are concerned with currently (discrimination in employment cases), one has the Vento guidelines which divide successful cases into three bands, that inform the amount of compensatory damages. In practice hence, a court will consult statutes, guidelines, and past relevant cases, to decide which band a case falls under and hence how much compensation is due. It should be noted that each band stipulates a certain range the damages must fall into and that the exact amount won in a particular case remains up to the court to decide.

This procedure suggests an obvious classification problem which is that beginning with the claims of an aggrieved party whether it is possible to predict which band their case would land into **should it be successful**. It is important to keep this proviso in mind because the claims are susceptible to the parties' own biases, and the court may not find in their favour. For the problem statement, this means that we are not interested in predicting whether a given case will be successful but the compensatory damages awarded, should it be successful. In this paper, we will propose a few different methodologies to tackle this problem. The first few will be simpler to implement and provide reasonable results in certain cases, even when working with a small dataset. More advanced methodologies based on Machine Learning and Neural Networks will be proposed but an effective implementation will not be possible to showcase in this report, for reasons to be discussed in depth later. The methods proposed in this report are relatively simpler to implement but in the final section, we will recommend some of the state-of-the-art learning ensembles that we expect to work very well for this project.

1.1. Data

It is well known that the strength of a well-coded learning algorithm depends on the quality and quantity of the data it is trained upon. What we mean by quantity is obvious, but the quality of data is something that warrants explanation. By quality, we mean that the data we train the model is exactly the kind of data we want to use the algorithm on, while also boasting what one would call proper English syntax and grammar. In the best-case scenario, one would like to have a dataset comprising the claimant's allegation as they would report it to their lawyer labelled by the band they were eventually put into. Due to privacy concerns, this is not possible, which forces us to approximate the expected input. One manner of doing this would be by extracting claimants claims from case summaries or remedies retrieved from either the government or BAILII website. As we are looking for small summaries of the allegations of the claimant themselves, the summaries should

Email address: mnadeem@ed.ac.uk (Usama Nadeem)

not have any of the tribunal's findings, assessment of the claimant's reliability, the response from the respondents, etc. Another constraint is that of size. Thus post-extraction, the data would have to be cleaned and, if needed, summarised by hand.

For our work here we have a small database of 45 settled cases, comprising 11 lower band, 25 middle band, and 8 upper band cases, that have **not** gone through the preparation we have highlighted in the previous paragraph. We forewarn that due to the small size and lower quality of the dataset, proper implementation of the methods is not possible, and neither is it possible to validate the models being used here and report on their efficacy.

2. Similarity of syntax

We envision the problem as follows: given a certain dataset of case summaries each labelled according to the band that they fall into, we are to ascertain how to label a new unseen, unlabelled case summary. Due to the manner in which the bands are defined, we expect each summary to be more **similar** in some sense to other cases under its label than those under another label. For example, cases involving certain discriminatory behaviour might be more egregious and merit a higher band than those of a lesser slight and as such we expect the former sort of the cases to be **clustered** away from the latter ones. If one finds the new case summary to be more similar to cases in a particular band than any other, then one would be inclined to think that it will be ruled upon similarly. Measuring this similarity is a standard problem in Natural Language Processing and much progress exists on this front. The general prescription is to generate a **vector space** wherein the case summaries lie. Then each case summary can be represented as a vector and the problem of comparing case summaries is reduced to comparing vectors. Comparing two vectors is not particularly difficult; one can appeal to the cosine similarity as a measure of similarity:

$$S_c(\bar{A}, \bar{B}) = \frac{\bar{A} \cdot \bar{B}}{\|\bar{A}\| \times \|\bar{B}\|}$$

where $\bar{A}$, $\bar{B}$ are vectors, $\bar{A} \cdot \bar{B}$ denotes the usual dot product of the two vectors and $\|\cdot\|$ the associated norm. S_c is bounded between 1 and -1, with 1 occurring if and only if $\bar{A} = \alpha\bar{B}$ (which is to say the vectors are the same in direction) and -1 if and only if $\bar{A} = -\alpha\bar{B}$ (which is to say the vector are diametrically opposite) for $\alpha > 0$.

The more important question is how to relate a case summary to a unique vector. One of the simplest yet surprisingly useful way, is that of Bag-of-Words.

2.1. Cosine Similarity with Bag-of-Words

Under the Bag-of-Words paradigm [5], one first constructs a corpus of all the unique words in all the case summaries. Then each case summary is represented by a vector whose length is the number of words in the case summary, and the component of the vector is the number of times each unique word in the corpus is repeated. So for example if we have the corpus $\{\text{cat}, \text{dog}, \text{this}, \text{is}, \text{a}\}$, then the sentences "this is a cat" and "this is a dog" will be represented by the vectors $(1, 0, 1, 1, 1)$ and $(0, 1, 1, 1, 1)$. These two sentences are obviously very similar and the cosine similarity metric should be close to 1 (it is 0.75). One notices that this method does ignore grammar, subtext, and the context of the sentence; relying squarely on the words (or phrases depending on implementation) in the case summary. This makes this algorithm very easy to implement. It should also prove an effective choice for the problem at hand provided we have a large amount of varied data because one does not expect the grammatical nuances to significantly impact the court's judgement but rather the keywords that describe the discrimination. As we expect same keywords to be used (say harassment, unlawful termination, slurs, sexist remarks to give a few examples) in reporting similar cases, we expect this method to produce good results. Consider for example the following case summary:

"Following a TUPE transfer S was addressed by the respondent as 'Mr' on the telephone and, over a five-month period, on at least five occasions in writing and also on portal queries used by her following an administrative error. S, though found to be agitated by the respondent's error and caused anxiety and stress, did not raise the issue as part of a grievance. Her formal complaints were instead directed towards travel costs, though she did raise the discriminatory treatment verbally with two people including the grievance decision maker. The tribunal held that S had been subjected to harassment related to sex in that calling her Mr violated her dignity. The tribunal held that the award fell towards the bottom of the lower band of the Vento guidelines."

By running our algorithm we can attach to each case summary a vector. To see which band our case summary belongs to all we need to do is find the centroid (the geometric centre) of each band and calculate the cosine similarity of our test case and the centroids. We find a similarity score of 0.740 with the lower band cases, 0.734 with the middle band cases, and 0.724 for the upper band cases. This suggests choosing the lower Vento

band, which we know to be the correct band for this case. In this manner, one is able to check whether a case summary falls into a particular case or not. As warned before, we do not have enough data to perform any reasonable model validation at this point, but we do remark that as the underlying dataset grows, so will the classifying power of this algorithm. To make sure that trivial connections between articles and prepositions, and other lexical groups don't artificially inflate the score, one would also need to require certain character limits on the summaries that are admitted. Another way to ameliorate these disadvantages is to consider the alternative TF-IDF approach.

2.2. Cosine Similarity with TF-IDF

TF-IDF [5] is an approach that helps obviate some of the disadvantages that BoW suffers from. Namely, it **normalises** the vectors so that longer documents are not advantaged merely by virtue of being longer. It also adds weight to less frequently occurring words that are nonetheless important. Then one is able to do a similar sort of analysis as was done in the previous section with the new score: 0.471, 0.440, 0.456. It leads to the same conclusion as before.

In practice however, it is advisable to use both BoW and TF-IDF in succession for the best results. In this particular case, we will get the following figures: 0.471, 0.441, 0.457. While the improvement is minuscule, in general, when dealing with larger amounts of data, with each case summary of substantially different lengths, this method should perform better than the two individually

2.3. Similarity of Semantics

We saw in the previous section that the BoW and TF-IDF methodologies can be employed to calculate the similarity between sentences, according to the words that they share. While this can be an effective approach if we have a huge amount of variegated data, one desires a classifier that is able to acknowledge the similarity between the sentences: "The Prime Minister spoke in Westminster", and "Boris Johnson addressed the House of Commons". Our existing methodologies would fail to capture this similarity to any appreciable degree because "the" is the only word that is common to the sentences and hence we would expect the cosine similarity to be very poor.

The remedy to this is the approach of word (or document) embedding [6]. Here too we are looking to generate a vector space where each word (or paragraph) has a learned representation coming from an ML model. Pictorially, you can imagine each word (or document) to be a point in some n-dimensional space that retains the semantic features of that word (or document) by virtue of its position. For example, the trained model would keep the words "House of Commons" and "Westminster" close in space, and in this way, one can capture the similarity (or dissimilarity) that originates from synonyms (or antonyms). Even more is true of these learned models, that we eschew for the present exposition, but what is relevant is that similarity now can now be calculated simply by calculating the Euclidean distance of their vector representations:

$$d(\bar{A}, \bar{B}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

where as before $\bar{A}$ and $\bar{B}$ are vectors, and a_i and b_i denote their respective coordinates.

It is possible to train such a document embedding for one's data with Word2Vec (or Doc2Vec for our work here) developed at Google. The benefit of training it on our dataset is that it will be able to capture the underlying contextual, and legal intricacies to allow for comparing case summaries even if the same words are not directly used. This methodology has seen great success in word vectorisation but we caution here again that for this methodology to give a good result one would need a large dataset for it to work. Once one trains a document embedding over the entire dataset of the case summaries, one can extract vectors for those case summaries and thence for newer case summaries, we can **infer** vector representations. Then by checking the Euclidean distance between two vectors, we can judge how close they are. For our running example so far, we can work out the following distances between the case summary and the centroid representations: 3.901, 3.96, 3.918. As we are looking for minimal distance here, we will choose the lower band for our case summary.

2.4. BERT

While Word2Vec (and the derivative Doc2Vec) are shown to be more flexible than the BoW methodology, Word2Vec still suffers from the fact that it attaches fixed vectorisations to words. Consider the admittedly clunky sentence, *They made for the river bank, after robbing the bank*. Under the Word2Vec paradigm, both instances of the word *bank* would be represented using the same vector even though we know that the words carry different meanings. To deal with these ambiguities, we can use BERT [4], again from Google, to carry out the vectorisation of the data. BERT is the state-of-the-art in

language representations and is used across the industry for entity recognition, question answering, and of course classification. To check the similarity we would again calculate the centroid and take the Euclidean distance. **spaCy** library for Python allows for very easy implementation of this methodology.

3. ML Classifiers

3.1. Naive Bayes' Classifier

We have till now seen ML methods used to construct the vectorisations of the data but the rule for classification was deterministic, usually the optimisation of some (dis-)similarity metric. We now propose our first ML classifier - the Naive Bayes' Classifier [8]. These classifiers rely on the Bayesian statistical notion of a posterior distribution, and the *naive* assumption of independence upon the inputs. The probabilistic theory behind the classifier is immaterial, but what is important is that again we need to translate our case summaries into numbers via vectorisation. Here the BoW followed by TF-IDF is the recommended vectorisation because for finite labels, one uses the Multinomial Naive Bayes and that assumes positive features (the components of our vectorisation) which rules out Doc2Vec or BERT. The implementation is not difficult, as the usual python package **Scikit-Learn** carries an optimised version of the classifier that is easily applied to the vectorisation one would get from the BoW (or TF-IDF) method above.

3.2. Support Vector Machines

The next ML methodology we propose is that of Support Vector Machine [3]. This methodology much like the similarity arguments we made in the initial sections, is geometric in nature. Again one employs some vectorisation method to envisage the case summaries as points in space, but now instead of minimising any metric, one endeavours to fit an optimal hyperplane such that it separates the entire n-dimensional space into smaller spaces containing the different classes. Then for an unseen point, the problem is merely checking in which smaller space it lies and labelling it correspondingly. One benefit this has over Naive Bayes' is that we no longer have to worry about the positivity of the coordinates of the vector representations, and hence we don't have to exclude the high-performing vectorisation methods like BERT. Implementation of SVM too is made extremely easy by the use of **Scikit-Learn**.

3.3. Random Forest

One of the most popular ML methodologies employed in classification and regression problems is that of Random Forests [2]. This involves constructing a large number of decision trees from bootstrap samples from the training dataset. While an implementation from scratch is possible, it is again made very easy by the existing Scikit-Learn library. To implement this methodology one needs data (our case summaries here), some fixed number of input features (which for us will come from the vectorisation) and finally the classes (lower, middle, and upper) to train the classifier.

3.4. Multi-layer Perceptron

The final proposal we make is a quintessential example of a Neural Network supervised learning methodology. Under this paradigm, one is trying to learn a function $f : \mathbb{R}^m \to \mathbb{R}^n$ by training on some dataset, that will function as our classifier [7]. For our case, we will chose $n = 1$, in particular we will use the subset $\{1, 2, 3\}$ with the understanding that they correspond to the set {lower, middle, upper} respectively. m will be decided by the vectorisation of the data that is being used, so for example if a particular lower band case summary has the vectorisation $(a_1, \ldots, a_{20})$, then f will be such that $f(a_1, \ldots, a_{20}) = 1$. For an unknown summary, whatever f maps the vectorisation to will be its predicted class. A first-principles implementation of the Perceptron is not simple, but again there exists a library that can take care of this for us. The only thing to be decided upfront is what vectorisation is to be employed.

4. Stacking Classifiers

We have up till now proposed a variety of classifiers based on a variety of classifying principles to be used towards solving this problem. While each of the classifiers proposed have been proven both theoretically and practically to perform well, one can often increase the classifying power by *stacking* different classifiers. The idea is to combine different classifiers by means of a meta-classifier. The implementation can be difficult but it is made easier by the use of existing libraries. For this problem, we can use a combination of Support Vector Classifier (Sec. 3.2), Random Forest Classifier (Sec. 3.3), and Multi-Layer Perceptron (3.4), and use the library **mlxtend**.

5. State-of-Art

We intimated in the beginning that the methodologies we propose in this report are tested, standard methodologies that have been and are still being used in the industry for various classification and regression problems. I expect that these will perform very well for the problem we have at hand (my own testing has had positive results). That being said the State-of-the-Art in this field has progressed quite a bit. We gave one example in the form of BERT and pointed out the advantages it offers over traditional methodologies of BoW and TF-IDF, but even BERT has been recently outperformed. **Multi-Task Deep Neural Networks for Natural Language Understanding**, is one example of such an algorithm. This suggests that while this problem is certainly tractable to standard Machine Learning solutions, more advanced methodologies might be investigated for more accurate performance when an implementation is effected. If I was personally coding this, I would like to explore the new advances made in Recurrent Neural Networks (RNN) by Akbik et al. [1], for which the code is publically available. This is the state-of-the-art in NLP, and is what I would use to get the vector representations that are required to make any of classifiers in this report work. To apply Flair, one could use the pretrained transformers (like BERT) to get the vector representations directly or make use of its Flair Embeddings to construct word embeddings for our own dataset of case summaries that are capable of capturing syntactic and semantic nuances and then training an RNN over it to recover sentence embeddings. Needless to say that the former is recommended if the training set (our dataset of case summaries) is small. With this vectorisations any of the classifiers that allow for negative vector components given in this report may be applied, but I would recommend using the classifier that comes with this algorithm. This classifier is meant to perform well with very small training sets. In some general tasks, like sentiment analysis where one wants to classify a text into either good or bad, no training set was required at all to effect a very potent implementation. For this reason, we expect the Flair classifier to perform very well for this problem, where we may not have a lot of quality data.

6. Conclusion

In conclusion, we would like to contend that a ML solution to this problem is very feasible. At the heart of it, this problem is a classification one, for which ML methodologies are being use Industry-wide to great success. In the following list we collect some reasons why we expect this problem to be amenable ML algorithms:

- By the nature of the common-law system, one expects there to be similarity between cases falling into the same band. ML methods are great at picking up these similarities.
- As we expect similar vocabulary to be used to explain similar sort of cases, even older, easier to implement methodologies (like Cosine-Similarity with BoW for example) can be very effective, where for general text classifications they are known to be somewhat poor.
- Amongst the banes of NLP programmes are irony and sarcasm. Machines, in general, expect words to connote what they mean, and hence struggle with sarcastic comments. As we expect (or can at least enforce at the stage of data input) formal recounting of the claimant's allegations only, our methods should work well.

There are some still some possible pitfalls:

- The quality of the data on which the models are trained on, is the single most important concern. For best results one would want to train it on the same data we want to work it on, that is claims directly from the aggrieved that have also been labelled by the Vento band they fall into. This due to various concerns (privacy for one) is not possible.
- With approximate input data (we take the claims from official judgments in this report) one has to be very careful to curate it so that it it emulates closely the aggrieved's claims, otherwise we run the risk of the machine learning the "wrong things".
- The quantity of data is also a concern. As a general maxim, one needs a lot of data to make ML methods work reasonably well. Often this means, data to the tune of hundreds of thousands. In our "claims extracted from judgments" models, we have data only to the order of a few hundred. Even the Flair classifier that works well with low quantity of data would undoubtedly perform better with more data in the training set.

References

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 08856125. doi: 10.1023/A:1010933404324. URL http://link.springer.com/10.1023/A:1010933404324.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. volume 20, pages 273–297. Springer, 1995.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[5] Yoav Goldberg. Neural network methods for natural language processing. Synthesis lectures on human language technologies, San Rafael, Calif., 2017. Morgan Claypool Publishers. ISBN 9781627052986.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[7] Daniel A. Roberts, Sho Yaida, and Boris Hanin. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, 2022. doi: 10.1017/9781009023405.

[8] Harry Zhang. The optimality of naive bayes. volume 2, 01 2004.