A model for predicting topic relevance

Bill Oxbury

April 2023

This note details the model used by BirdLife International's *LitScan* project for scoring scientific articles for conservation relevance, and makes some comparisons with approaches that use binary classification.

In section 1 we set up the general setting of document classification, and in section 2 notation and terminology for TF-IDF vectors, a key tool for this setting.

In section 3 we describe the problem of finding documents of relevance to a given topic (in our case avian conservation), and relate it to the context discussed earlier. We describe the *LitScan* model — conceptually in section 4, and practically in section 5.

In section 6 we discuss the alternative approach of training a binary classifier, as in [1]. In particular, we outline reasons to be cautious of this approach — but in section 7 we outline a way to derive a binary classifier from the *LitScan* model in order to facilitate a direct experimental comparison.

1 The problem of characterising documents

In this note we'll refer to text documents d drawn from some *corpus* of documents D. Each document d is a sequence of *tokens* (or *words*) w belonging to some big dictionary of words W.

A documents d may be full scientific articles, dissertations etc; or they may be title-plus-abstract of such articles; or they may be single sentences.

The words w of which d is composed are assumed to have been preprocessed: with punctuation and common stop-words (*the*, *and*, *if* etc) removed and the remaining words normalised to word stems (e.g. *test*, tests, testing all normalised to *test*). This is the sense in which a document is treated as a formal sequence of word tokens.

Although the current experiments are in English, we do not need to make any assumption about language. The corpus D may consist of documents in any written language.

Given such a corpus of documents, there are two kinds of problem we might be trying to solve:

> *Problem 1: what different sorts of document are in D?* How similar or dissimilar are documents with each other? Are they all of interest or just a subset? Do they cluster into recognisable topics?

For example, D might consist of all articles published by *bioRxiv* over some period of time. We are primarily interested in avian ecology, say, but we would like to understand the 'big picture' of what's in our *bioRxiv* corpus, and then use that knowledge to locate articles relevant to avian ecology within the corpus.

> *Problem 2: characterising documents of D in order to find similar documents elsewhere.* We want to be able to recognise when documents *outside* of D are similar to those in D. How do we characterise D in the universe of all possible documents?

For example, D might consist of IUCN assessment reports on avian ecology and population trends. This is our topic of interest, and we would like to use a model of D to help us find articles relevant to this topic across a wide range of academic and other sources.

2 Term frequency and document frequency

This section describes methods well designed to answer Problem 1.

We have a document corpus D with words (= normalised tokens) drawn from a dictionary W. Given a word $w \in W$ and document $d \in D$, the *term frequency* is

$$\mathrm{tf}(w,d) = \frac{\text{number of times } w \text{ occurs in } d}{|d|}, \tag{1}$$

where $|d|$ is the size of d, i.e. the total number of words. So $0 \leq \mathrm{tf}(w,d) \leq 1$ and we can consider the vector

$$\mathrm{TF}(d) = (\mathrm{tf}(w,d))_{w \in W} \tag{2}$$

of length $|W|$. This is zero except at the words in d, and its nonzero entries are the proportions with which words occur in d.

The second vector we are interested in depends on the whole corpus D and not on individual documents. This is

$$\mathrm{IDF}(D) = (\mathrm{idf}(w,D))_{w \in W} \tag{3}$$

where

$$\mathrm{idf}(w,D) = -\log\left(\frac{\text{number of documents } d \in D \text{ containing } w}{|D|}\right), \tag{4}$$

called the *inverse document frequency*.[1] It can be thought of as (minus log of) the probability that a random document from D contains the word w. The vector $\text{IDF}(D)$ has large positive weight at words $w \in W$ that are rare in the corpus D, and low weight at common words. The weight will be zero at words that occur in all documents.

Let's get back to Problem 1 in the previous section. As a way to explore the documents $d \in D$, it's natural to characterise them by their word distributions, i.e. by their TF-vectors $\text{TF}(d) \in \mathbb{R}^{|W|}$. Even better, in this vector representation we can down-weight the common words, and accentuate the effect of the more unusual words, by multiplying term-by-term by $\text{IDF}(D) \in \mathbb{R}^{|W|}$. This results in the *TF-IDF vector*:[2]

$$\text{TF-IDF}(d) = \text{TF}(d) \odot \text{IDF}(D) \in \mathbb{R}^{|W|} \tag{5}$$

This vector gives a strong signal to discriminate documents in the corpus having significantly different word distributions, and is commonly used as a feature vector for both unsupervised methods (e.g. topic modelling) and supervised methods (e.g. binary classification for labelled documents from the corpus — including in the paper [1]).

3 The LitScan problem

BirdLife International's *LitScan* project (unpublished, but described briefly in the blog [2]) is an example of Problem 2. It starts from data consisting of red-list assessments for 11,107 bird species, and asks how to identify articles in the wider scientific literature that are relevant to the task of growing and updating these assessments.

Here are some observations about the data. As always (see section 1) the assessment texts are first cleaned up (with things like citations removed) and presented as tokenised sentences. In fact, our approach (in the next section) will be to take documents D to be the individual sentences, rather than the full texts.

In total, the texts contain a total of $|D| = 91{,}070$ distinct sentences, drawing on $|W| = 31{,}489$ words. Figure 1 shows that the word count and sentence count for an assessment are well correlated — or in other words that sentence lengths are reasonably constant at around 24 words.

Figure 2 shows the behaviour of the sentence count and in particular how it depends on the red-list status of a species: typically, the, more vulnerable a species, the more is written in its assessment.

[1] In this definition, we can add +1 to the numerator to prevent infinities at words $w \in W$ that do not occur in any document. But to keep things simple, we'll just assume that all words appear in some $d \in D$.

[2] The symbol $\odot$ denotes termwise muliplication — this results in a vector of the same length as the factors.

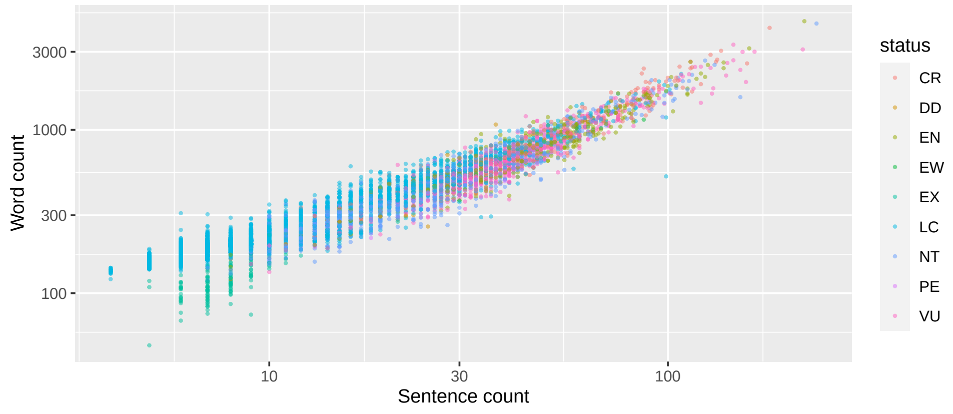

Figure 1: Red-list assessment texts for 11,107 bird species, plotted by word count against sentence count and coloured by IUCN status category.

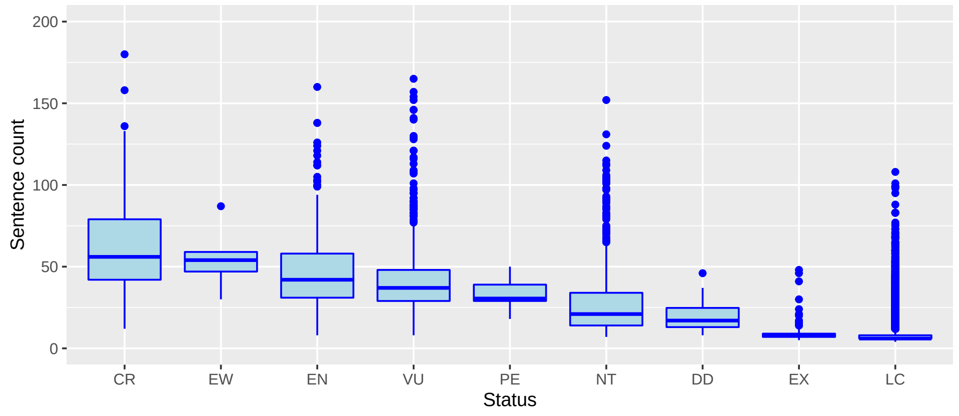

Figure 2: Another view of the data of Figure 1: here sentence count is plotted against red-list status.

4 The LitScan model

If our task were Problem 1 (section 1) — analyse the red-list assessments — then a natural approach would be to take as our corpus D the set of 11,107 assessment texts (suitably tokenised) and represent them by their TF-IDF vectors. But this is not our task — we are less interested in discriminating among assessment texts, and more interested in discovering scientific articles whose text 'looks like' that of the assessments. This is Problem 2.

So our strategy is a little different. We will take as our corpus D the set of 91,070 distinct sentences in the assessment texts. And we will make use of the single vector $\mathrm{IDF}(D) \in \mathbb{R}^{|W|} = \mathbb{R}^{31{,}489}$, which represents the overall word preferences of sentences in the corpus. How can we use this vector to score new documents, taken from the scientific literature, for relevance to us?

Suppose x is a new document — say the title-plus-abstract of some journal article. Recall that its TF-vector (2) encodes its word frequencies. We will define

$$R_D(x) = -\mathrm{TF}(x) \cdot \mathrm{IDF}(D), \tag{6}$$

and call this the *relevance* of x to the corpus D (which in our case is the set of red-list assessment sentences).

(6) is a dot product of two very long vectors. However, $\mathrm{TF}(x)$ is zero everywhere except at the words of x, so $R_D(x)$ is only a small summation. And it follows from (1) and (3) that it can be expressed as an average over the words of x:

$$R_D(x) = -\frac{1}{|x|} \sum_{w \in x} \mathrm{idf}(w, D) \tag{7}$$

Comparing this with (4), we can interpret $R_D(x)$ as the average log-likelihood, over words of x, of being found in documents (i.e. sentences) of D. $R_D(x)$ is always negative, but the closer it is to zero, the more relevant is the document x, while the further x is from the topic of D, the more negative $R_D(x)$ becomes. If x contains any words that are not in D at all then $R_D(x) = -\infty$.[3]

5 The model in practice

So how does the *LitScan* model work in practice? The model is entirely defined by the IDF-vector (3). This is computed once for the corpus D of cleaned-up sentences from red-list assessments, and stored in a JSON file.

At query time we have a document x which is the title-plus-abstract of some scientific article. The relevance score of x is simply computed from the formula (7).

[3]In practice, *LitScan* sets $-\infty$ to a finite value -20.0. But the fact that D consists of more than 91,000 sentences and 31,000 words mean that our model has good word coverage.

-3.781	*Habitat loss pushing more bird species to near extinction*
-4.171	*Evidence of widespread declines in Kenya's raptor populations over a 40-year period*
-5.000	*The Black Noddy breeding population at Heron Island, Great Barrier Reef*
-6.000	*Diversity, endemism and conservation issues of the avifauna of Tunari National Park*
-7.000	*Insectivorous bird communities of diverse agro-ecosystems in the Bengaluru region*
-8.000	*Telomere length and dynamics of spotless starling nestlings depend on nest-building materials*
-9.000	*Phylogenetic relationships within the Alcidae inferred from total molecular evidence*
-10.000	*Molecular characterization of cryptic haemoproteids in the laughing thrushes*
-11.001	*Hypodectes propus in a rufous turtle dove*
-12.003	*Sequencing and analysis of the complete mitochondrial genome of Podoces hendersoni*
-13.004	*Seed predation by parakeets Brotogeris chiriri*

Figure 3: Example titles at descending values of the relevance score (left-hand column)

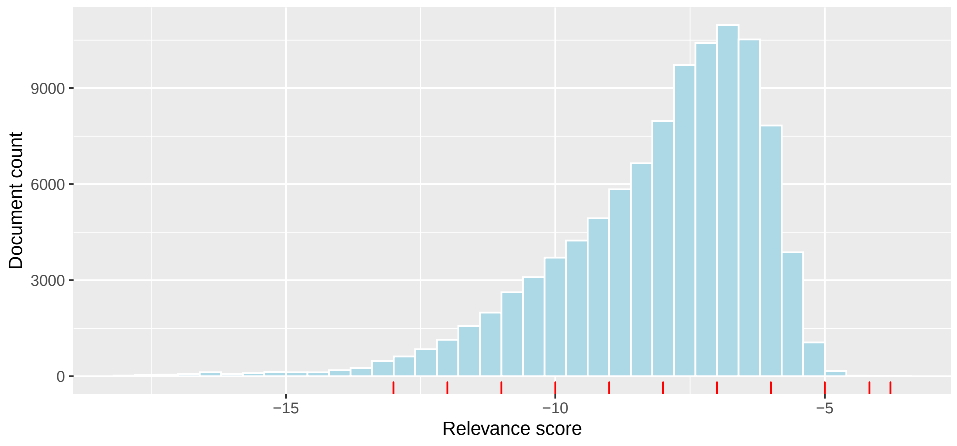

Figure 4: Histogram of the relevance score over 101,594 query documents. The red ticks are at the values of the sample titles shown in the table above. The median is at −7.637 and upper quartile at −6.669.

For 101,594 such query documents in the *LitScan* database as of this writing, Figure 4 shows the distribution of this score. Example titles have been sampled at the red marks — basically at scores as close as possible to integer values, just for illustration. These sample titles are shown in Figure 4. The top score -3.781 is the highest currently in the database. Below this value, the titles give a sense of the descending relevance to red-list assessment. (Of course, the reader should keep in mind that relevance is computed on the abstract too, which is not shown here.) The upper quartile at -6.669 seems a good threshold above which to expect high relevance.

6 Binary classification

A supervised learning approach to detecting conservation relevance has been taken elsewhere [1]. The problem is treated as binary — a query document is either relevant or it is not relevant. A training corpus of titles-plus-abstracts is constructed, with relevant/irrelevant class assigned in equal numbers, and binary classifiers (logistic regression and convolutional neural network) trained on this data set.

In this section we will discuss some cautions to be aware of in taking such an approach. Binary classification of this sort has some pitfalls which *LitScan* avoids. Nevertheless, it would be desirable to make a quantitative side-by-side comparison, and in section 7 we will suggest how to do so.

Our first caution on the supervised approach is the choice of negative ('irrelevant') texts. In [1] these are sampled from one specific data source (the National Center for Biotechnology Information). But this choice will constrain the resulting model, and it is not clear how much variation would be induced in the learned classifier by making other choices of negative training points.

To illustrate this, Figure 5 shows the result of applying logistic regression to three toy training data sets in 2 dimensions. They all have the same positive training points (black) but different choices of negative points (blue). In each case, the learned classifier is the linear decision boundary in the space (blue line).

The choice of training data affects the model in a second way too. The feature vector on which logistic regression is trained is (at least for the model LR-A in [1]) the TF-IDF vector of a document. But recall that this depends on the training corpus D (see (5) and (3) in section 2), and in particular on the choice of negative training data points. In other words, the weighting of the feature vector components is implicitly biased by the linguistic usage in the corpus D.

A third caution has to do with the size of the training data sets. This is inevitably constrained by both availability of suitable data and any human effort that's needed to label or verify labels. Manual construction of training data imposes a significant cost.

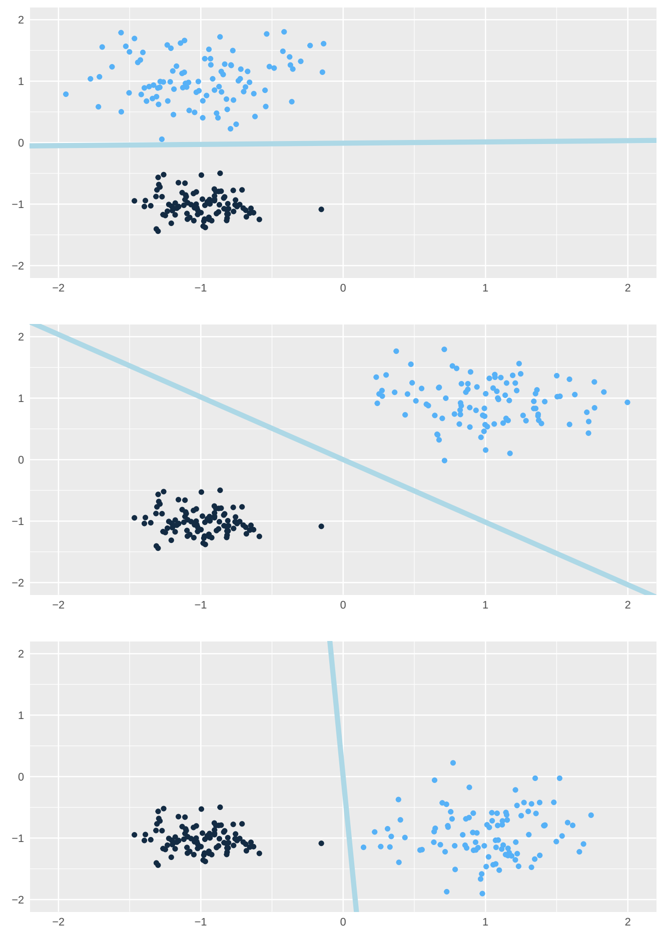

Figure 5: Logistic regression on 3 training data sets. They all have *identical positive* training examples (black), but *different negative* examples (blue). Even in 2 dimensions, the resulting decision boundary is highly sensitive to the choice of negative examples.

In [1], two training data sets are constructed, with equal numbers of positive/negative points, of sizes 1,266 (LPD) and 1,072 (PREDICTS). Are these sufficient, and how much training data is needed?

Suppose we are training a logistic regression in N-dimensional feature space. This means we are learning $N + 1$ coefficients of a linear decision boundary (as in Figure 5 for the case $N = 2$). A single data point imposes at most one constraint, and so it is evident that to learn a robust model we need at least N training points and preferably $\gg N$.

In the case of TF-IDF vectors, $N = |W|$, the number of word tokens in our dictionary. In [1], we could not find an explicit statement of the size of W, though it is presumably many thousands (and the figure of 400,000 is given in the context of word vectors for the CNN classifier). In this setting, the training sets of size $\sim 1,000$ seem unlikely to be sufficient. For training nonlinear CNN classifiers, the situation is much worse, as the learned decision boundary is much more complex than linear.

LitScan is designed not to suffer from these problems: there is no choice of negative training points needed; the training data that is used is a large source of (91,000) 'naturally occurring' sentences; and the model *is* the IDF vector of these sentences — no further coefficients are learned. Nevertheless, that does not prove it is a better model! In the next section we will discuss how to make a like-for-like comparison.

7 Direct comparison

We would like to make a direct comparison of the performance of *LitScan* with the supervised models of [1]. That is, we would like to test both *LitScan* and the classifiers of [1] on a suitable labelled data set — possibly derived from the data described in [1], though distinct from the data used to train their models.

LitScan outputs a continuous relevance score, but for the purpose of this experiment it is easy to use that to build a binary classifier, as follows.

LitScan consists of a relevance function $R_D(x)$, constructed from a corpus of sentences D and evaluated on a test document x (see (7) in section 4). To treat this is a binary classifier, we set a suitable threshold $\theta \in \mathbb{R}$ and we say that the output of *LitScan* is 'relevant' if $R_D(x) > \theta$, otherwise 'irrelevant'.

That leaves two questions: what corpus of sentences D to use for R_D; and how to set θ?

For D we have two choices. We could stick with the same sentences as used to train the BirdLife *LitScan* . Then the model is ready to use off the shelf. The downside of this is that it is focussed on avian conservation, and this may not be suitable for the test sets used in [1]. The alternative would be to construct a similar-sized sentence corpus based on the data of interest to [1].

So assume that the relevance function R_D has been built, and we need to learn θ for our binary *LitScan* classifier. For this we use the same or similar training data to [1]. This will consist of pairs (x, y) where x is a training title-plus-abstract and $y = \pm 1$ is its relevance label. We set

$$\theta = \underset{\theta'}{\operatorname{argmin}} \sum_{(x,y)} ||f_{\theta'}(x) - y||. \tag{8}$$

Here $f_{\theta'}(x)$ means the threshold classifier: $f_{\theta'}(x) = +1$ if $R_D(x) > \theta'$, otherwise $f_{\theta'}(x) = -1$. The equation (8) can be solved numerically just by computing the sum explicitly over a discrete range of values of θ'. (See Figure 4.)

References

[1] Richard Cornford, Stefanie Deinet, Adriana De Palma, Samantha L. L. Hill, Louise McRae, Benjamin Pettit, Valentina Marconi, Andy Purvis, and Robin Freeman. Fast, scalable, and automated identification of articles for biodiversity and macroecological datasets. *Global Ecology and Biogeography*, 30(1):339–347, 2021.

[2] Bill Oxbury. Language barriers in global conservation. https://billoxbury.github.io/environment/language_barriers/, 2023.