3.2. Case study 1: Naïve Bayes and the language of partisan news

3.2.1. Overview

Against the background of a degree course in Speech and Language Processing, the author sought to build a text classifier that would label news media with respect to its veracity and reliability. This labelling was to rely on stylistic, or lexico-syntactic features while using established Natural Language Processing (NLP) methods and supervised machine learning systems. Supervised machine learning involves training and test phases, wherein a program learns the transformations that produce known output-label c_i from input x_i, and then applies these transformations to new data to assess the probability that label c_i is applicable, or not. In terms of the learning architecture, the author used a *Naïve Bayes* algorithm and a *Term Frequency* (*tf*) vectorizer. These were instantiated in the *Python* language using the *scikit-learn*[1] library for machine learning functions and the *spaCy*[2] library for NLP and machine learning functions. This architecture is explained fully in the following sections.

My overriding goal in Case Study 1 was to evaluate *need*, *trustworthiness* and *practicality* in general, but in particular:

a) *transparency* of detection/classification method and intelligibility of results

and

b) *accuracy* of results against a human-annotation benchmark

These are considered *trust-building factors*.

With respect to goal A:

Firstly, the Naïve Bayes algorithm at the centre of the tool is centred around probability. In the simplest terms, Naïve Bayes classifiers predict the probability of a label given past and present evidence. This section will interrogate Bayesian probability theory as a means of accurate and transparent decision-making.

With respect to goal B:

The primary measure of such a tool's efficacy is its accuracy rate (this is borne out by the literature in general; see for example Potthast et al (2018), Horne et al (2016), Ott et al (2013)). The author's hope was to maximise accuracy without sacrificing the *generality* of the model. When a text

[1] *scikit-learn* is a Python module for machine learning.
[2] *spaCy* is an open-source software library for advanced Natural Language Processing. See https://explosion.ai/blog/introducing-spacy

classifier learns the style or styles of a particular dataset too well, it is less able to cope with new input that is exogenous to the original data. This problem is called *overfitting*.

The fundamental questions that Case Study 1 set out to answer were thus: "How do machines classify text" and "How accurately do they classify against a pre-annotated human benchmark". With respect to both aims it was particularly important to establish what *features* are used to flag unreliable news and which of those features are the most *informative*.

All these concerns are addressed in the following sections.

The entire code, implemented in the *Python* language and in the *Spyder*[3] integrated development environment (IDE), is included in the appendix to this dissertation.

3.2.2. Dataset

Compiling and organizing suitable datasets is a significant challenge for the task of mis-/disinformation detection. Rubin et al (2015) have proposed a detailed set of requirements for a fake-vs.-real news corpus (see Table 3 on next page).

Using Rubin et al's proposal as a guideline, the author chose a carefully compiled, structured and annotated dataset, the *BuzzFeed-Webis Fake News Corpus*[4]. The Weimar-based *Webis* researcher team of Potthast et al (2018) collaborated with the U.S. media and entertainment website *BuzzFeed* to create the study *A Stylometric Inquiry into Hyperpartisan and Fake News* (the results are discussed in the Literature Review of this study, Section 2.3.1.)

For this dataset, *BuzzFeed* employees collated news articles from three mainstream U.S. news outlets (*CNN*, *ABC*, *Politico*) and six "prolific", so-called "hyperpartisan" (far-left, far-right) publishers (*The Other 98*, *Addicting Info*, *Occupy Democrats* (all left), *Eagle Rising, Freedom Daily*, *Right Wing News* (all right)) across a one-week period leading up the 2016 American election (Potthast et al, 2018, p. 233). Each article was labelled by the BuzzFeed content checkers as "mainstream" or "hyperpartisan". This forms the binary classification data-split for the machine learning tool in this case study.

The present study took a simplified text-body-only subset of the article data in the *BuzzFeed-Webis Fake News Corpus.* The author dispensed with metadata such as URLs that were in the original XML files. This is because the aim was to conduct a purely stylistic analysis.

[3] Spyder is an integrated development environment for the *Python* programming language for the purposes of scientific programming. See https://www.spyder-ide.org/.
[4] The dataset is publicly available at https://zenodo.org/record/1239675#.XLiC3-hKjIU.

The word counts were as follows:

“Hyperpartisan” News Corpus (HNC): 336,301 words, 819 texts

“Mainstream” News Corpus (MNC): 551,633 words, 808 texts

Total: 887,943 words, 1,627 texts

Table 3: Rubin et al’s (2015) criteria for a mis-/disinformation corpus applied to the *BuzzFeed-Webis* dataset

<u>Criteria for a mis-/disinformation corpus</u>

Availability of both truthful and deceptive instances

“Mainstream” sources are mostly objective. “Hyperpartisan” sources are mostly misleading or biased.

Digital textual format accessibility

Files are in XML format.

Verifiability of “ground truth”

Articles are fact-checked and checked for bias by BuzzFeed journalists.

Homogeneity in lengths

All texts are approximately 1,000 words in length.

Homogeneity in writing matter

Mainstream texts are mostly objective. Hyperpartisan texts are mostly hyperbolic/polemical.

Predefined timeframe

One-week period before the 2016 U.S. presidential election.

Pragmatic concerns

No particular pragmatic concerns.

Language and culture

All examples use American English and deal with U.S. politics.

adapted from Rubin et al (2015, p. 7)

3.2.3. Naïve Bayes classification and vectorized representations

3.2.3.1. Bayesian thinking and naïve algorithms

The model makes use of a Naïve Bayes classifying algorithm, which is considered an excellent (and fast) baseline model for binary textual classification tasks (Müller, Guido, 2016, p. 68-70) and a highly efficient algorithm for machine learning and data mining generally (Zhang, 2004).

Bayesian models are a class of subjective[5] probability classifiers that make use of known events to ascertain the probability of an uncertain but related event. In more formal terms we can call this:

the probability of *example E* having a label *c*, given that *E* has attributes ($x_1, x_2 \ldots\ldots x_n$) (adapted from Zhang 2004, p.1).

For this study, the *examples* are news articles, the *labels* are "mainstream" and "hyperpartisan", and the *example attributes*, which are henceforth referred to as *features*, are words or word patterns (unigrams and bigrams). For the purposes of machine learning and the training of the machine learning program, label *c* is *known* in each case due to the annotation work of *BuzzFeed* journalists.

A common simplification involves making the algorithm *naïve*[6]. In a naïve assignment of probability, all features are independent of one another within each class (here: "hyperpartisan" and "mainstream"), so that each feature contributes to the final classification in the same way regardless of which other features are also present. The naïve component of the classifier renders resource-consuming dependence-based calculations unnecessary, allowing many modern textual classifiers to operate with a set of around 10,000 terms (Flach 2012, 9).

The following is a run-through of the naïve classification process:

[5] Bayesian probability is *subjective* because, unlike *classic probability* (where probability is based on the assumption of equal chance of a set of events occurring, such as in the case of a 6-sided dice) and *frequentist probability* (where probabilities represent the long-run frequency with which events occur) it does not derive probabilities either purely from assumed a priori properties of the phenomenon nor purely from long-run observation but instead but offers a "degree of belief" (Levitin,2016, 99).

[6] For detailed algebraic representations of Bayesian and Naïve Bayes theories see Zhang 2004.

The classifier computes a posterior probability score for a label on the basis of the training data as follows: A statistical analysis shows how often a word (or *feature*), such as “Hillary”, occurs in documents with each of the given labels (“mainstream” or “hyperpartisan”). If “Hillary” occurs in 40% of “hyperpartisan” documents but only in 30% of “mainstream” documents, then for a document containing the term “Hillary” the posterior probability of that document being “hyperpartisan” is increased by a factor of 4/3. Bird, Klein and Loper (2009) describe this as a process whereby features “vote against” labels that they do not regularly co-occur with (p. 246).

At the level of real-world experience, Naïve Bayes seems counterintuitive and overly simplistic because the independence assumption is not mirrored in the highly interrelated phenomena we know. A given event or body of evidence is clearly contingent on other events and bodies of evidence. It must be said at the outset, that although the algorithm will assume words such as “kneel” and “national anthem” co-occur with no level of contingency, this assumption is clearly incorrect. The danger of this oversight is *double counting* during the test phase of the *train-test* machine learning process, wherein very similar or highly contingent features are counted and weighted individually in a way that is essentially superfluous. In such cases it may be expedient to use an algorithm that accounts for such dependencies (Maximum Entropy[7] *is one example,* see Bird et al 2009).

However, Müller and Guido (2016, p.68) state that it is Naïve Bayes’ holistic feature-inclusivity (the method of “looking at each feature individually”) that acts as the basis of its efficacy[8]. This is ultimately an issue of speed and simplicity vs. accuracy and complexity, where Support Vector Machines and Maximum Entropy Classifiers are likely to be slower and more complex to code but more accurate (Bird et al 2009, p. 251, Müller and Guido 2016, p. 68).

3.2.3.2. *tf vs. tf-idf* vectorizers

Once a classifier has been chosen, it is then necessary to represent the data numerically. Vectorization is the key step in feature generation for textual classification, whereby each feature is a *term* represented statistically as a (potentially weighted) *count* in the documents in a corpus. The literature on the subject (Müller and Guido 2016, Jarmul 2017) suggests two core vectorizing models for Naïve Bayes text classification: *Term Frequency* (*tf*) and *Term Frequency-Inverse Document Frequency* (*tf-idf*). Using the machine learning library *scikit-learn*, these representations are realised

[7] *Maximum Entropy* is a search-based algorithm that attempts to find the parameters that maximize the likelihood of a training corpus having a certain label. Parameters are initially random and then optimized iteratively (Bird et al, 2009, p. 250).

[8] For a far more detailed algebraic explanation of Naïve Bayes’ optimality (rooted in the distribution of dependencies) see Zhang (2004).

as vectorizers in certain ways. *scikit-learn's* "Count Vectorizer" is a simple raw-frequency-per-document count (a type of *Term Frequency*).

tf-idf builds on the initial *tf* method by performing an additional transformation. It offsets the simplicity of the word-frequency-per-document count by factoring in word occurrences across all documents in a corpus. If a word occurs many times in document X_1 but infrequently in documents ($X_2 \ldots. Xn$), it is likely to be very informative with respect to the style of document X_1. Such a word is assigned a higher weight by *tf-idf*. Conversely, a conjunction such as "and" that occurs frequently in all documents in a corpus, but is obviously not informative, is assigned a lower weight. The aim of *tf-idf* is essentially to arrive at words that are both frequent and distinctive markers (Nicholson, 2019).

tf-idf performed roughly 1 percentage point poorer than *tf* at an accuracy rate of 0.8157 and was thus discarded for the purposes of the study.

3.2.4. Linguistic intuitions

3.2.4.1. Hyperpartisanship

In line with Potthast et al's (2018) assertion that a style-based classification of "fake news" is not feasible (see Section 2.3.1.), the author took the approach of instead detecting extreme bias in news media (rather than veracity). However, it is important to note that samples taken randomly from the "hyperpartisan" dataset demonstrate both *bias* and *misleading content* that immediately call their veracity into question. This is borne out by the following claims from the far-right spectrum of the news dataset:

1) "They don't call CNN the Clinton News Network for nothing." (www.eaglerising.com)

2) "Al Sharpton, Louis Farrakhan and evil crony capitalist George Soros are also named as defendants in the filing." (www.rightwingnews.com)

3) "We are entering the black power phase of Black Lives Matter [sic] existence and it is crossing over into a terrorist organization. It's time to criminalize it." (www.breitbart.com)

All three claims above demonstrate a lack of objectivity. All three assertions are potentially defamatory[9] and are unlikely to hold up to any factual scrutiny. This demonstrates a correlation between extreme bias and misleading or false information. The author therefore proposes "hyperpartisanship" as an important *indicator* of *potentially false* news.

[9] Defamation is explained more fully in Section 3.3.3.

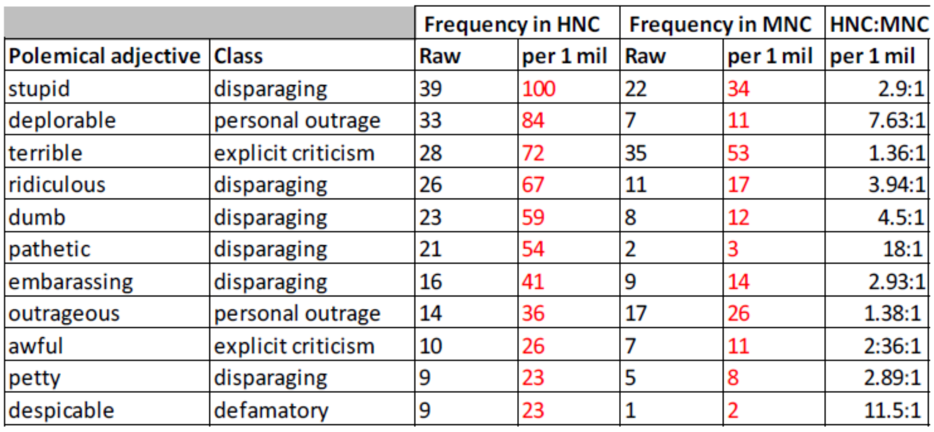
3.2.4.2. Lexico-syntactic insights

The author’s intuitions as to likely *lexico-syntactic insights* from the dataset were informed by two academic studies. Firstly, the surface-syntactic observations of Horne and Adali (2016), whereby “real” news articles are significantly longer than their “fake” counterparts. Fake news uses fewer technical words, shorter words, less punctuation, fewer quotes, and contains more lexical redundancy (p. 5).

Secondly, the author of the present dissertation had previously conducted his own stylistic study into disparate news types (Reid, 2019[10]), which demonstrated that “polemical” adjectives are a statistically significant stylistic feature when distinguishing between mainstream and far-left/far-right news media. Using the *SketchEngine*[11] corpus analysis tool, the author demonstrated that this type of adjectives occurs with far greater frequency in far-left and far-right news articles than in mainstream sources, and incidences in the latter class are often part of a quotation (Reid 2019, p. 10).

The following table from Reid (2019) shows raw frequency and frequency per-million-instances statistics for the “Hyperpartisan” News (HNC) and “Mainstream” News (MNC) corpora of the *BuzzFeed-Webis Fake News* dataset:

Table 4: Statistical comparison of instances of polemical adjectives



		Frequency in HNC		Frequency in MNC		HNC:MNC
Polemical adjective	**Class**	**Raw**	**per 1 mil**	**Raw**	**per 1 mil**	**per 1 mil**
stupid	disparaging	39	100	22	34	2.9:1
deplorable	personal outrage	33	84	7	11	7.63:1
terrible	explicit criticism	28	72	35	53	1.36:1
ridiculous	disparaging	26	67	11	17	3.94:1
dumb	disparaging	23	59	8	12	4.5:1
pathetic	disparaging	21	54	2	3	18:1
embarassing	disparaging	16	41	9	14	2.93:1
outrageous	personal outrage	14	36	17	26	1.38:1
awful	explicit criticism	10	26	7	11	2:36:1
petty	disparaging	9	23	5	8	2.89:1
despicable	defamatory	9	23	1	2	11.5:1

This dataset indicates that disparaging or defamatory language, or language expressing personal outrage or explicit criticism, occurs with far greater frequency in “hyperpartisan” news articles. The

[10] Available at https://www.researchgate.net/publication/335465465_A_stylistic_analysis_of_news_corpora_using_SketchEngine

[11] https://www.sketchengine.eu/

author's intuition was that such words would appear in sci-kit-learn's set of *most informative features*, where such features are the words or patterns that are most indicative of a certain label.

3.2.5. Preprocessing

Preprocessing of textual data primarily involved the removal of non-informative features using *spaCy* preprocessing parameters. These features included stop words (high-frequency function words such as conjunctions, articles, prepositions), punctuation and *named entities* (proper nouns). *Named Entity Recognition* (NER) was disabled[12] due to indications of previous studies (Reid 2019, p. 6, Culpeper 2009, p. 35) that proper nouns are not highly informative in differentiating between types of current affairs-based news corpora. This is because mainstream and partisan outlets both tend to deal with the same persons and organizations. Proper nouns are therefore not a *distinguishing feature*. With a view to future applications of the tool, it is also desirable to expunge era-specific named entities from a potential feature matrix. For example, "Trump" may not be a statistically significant token in partisan news media in 5 years' time.

The program also made use of basic tokenization (separating text into word tokens) and lemmatization (shortening words to base forms to avoid grammatical "double counting") schemes suggested in the *spaCy's* user manual.[13] n-gram patterns were restricted to unigrams and bigrams. Setting a minimum number of word occurrences (10 occurrences) removed obscure outlier words that could have skewed the outcome. Lower-casing to remove case-sensitivity is also a standard preprocessing step in the literature (Bird et al 2009, p. 45, Jarmul 2017).

Table 5: Overview of preprocessing with *spaCy*

n-grams	*unigrams and bigrams only*
POS limitations	*only adjectives, adverbs, interjections, nouns, verbs*
Lemmatization	removal of grammatical inflections such as plurals, conjugations etc.
NER	proper nouns disabled
Minimum word occurrence	*10*
Other	*stop words and quotation marks removed, promotional language*

[12] This step excludes proper nouns.
[13] https://spacy.io/usage/spacy-101#language-data

3.2.6. Machine Learning with *scikit-learn*

For the purposes of the machine learning process, the dataset is divided in a (random) 75%-25% split, whereby 75% of the data is used as a training set. The author performed a 5-fold cross validation on the data, which means the files were shuffled and split (75%-25%) five times in total. The process offsets the possible scenario in which the researcher "gets lucky" with the very first training-test division, in that the "hard-to-classify" documents are included in the training rather than test data (Müller and Guido 2017, p. 254).

scikit-learn uses several distribution models for its Naïve Bayes classifier. The author chose a *multinomial* distribution - recommended by Müller and Guido (2012) for larger documents - which uses the average value of each feature per label (Müller and Guido 2017, p. 69).

3.2.7. Results

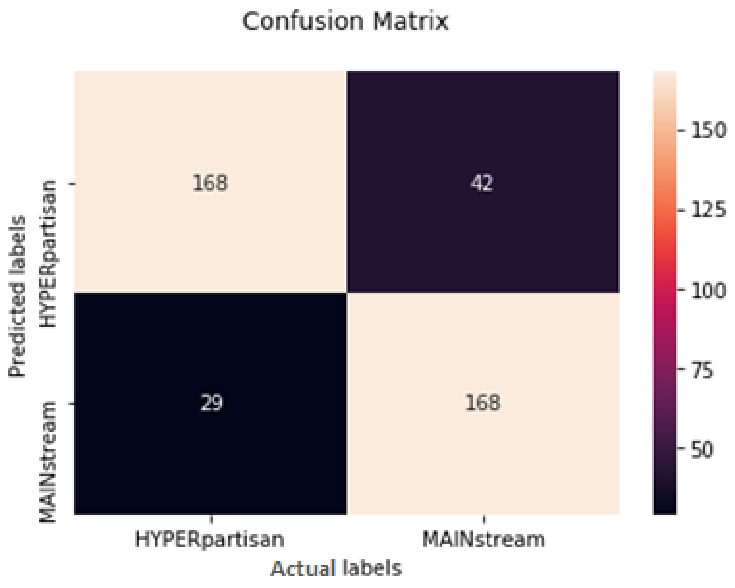
3.2.7.1. Accuracy metrics and informative features

A confusion matrix[14] (see Fig. 2 below) shows in detail how the machine learning model performed on the test data (407 articles; 197 "hyperpartisan", 210 "mainstream"). Rows represent *predicted* labels, columns represent *actual* labels. *Positive* cases are "hyperpartisan", *negative cases* are "mainstream". The matrix is divided into:

- 168 **true positives** (upper left square, "hyperpartisan" articles that were **correctly** labelled "hyperpartisan"),
- 29 **false negatives** ("hyperpartisan" articles that were **incorrectly** labelled "mainstream"),
- 42 **false positives** ("mainstream" articles that were **incorrectly** labelled "hyperpartisan"),
- 168 **true negatives** (lower right square, "mainstream" articles that were **correctly** labelled "mainstream")

[14] The confusion matrix was plotted using Matplotlib, a 2D plotting library for *Python* (https://matplotlib.org).

Fig. 2: Confusion matrix of classification results



The confusion matrix demonstrates that the program is slightly more efficient at handling "mainstream" classifications. The model had a 21% error rate for positive ("hyperpartisan") cases compared to a 14% error rate for negative ("mainstream") cases. The precision, recall and F1 scores[15] are shown in Table 6 below.

Table 6: Accuracy of model

Overall Accuracy	**0.82555**		
	Precision	**Recall**	**F1**
Hyperpartisan	**0.85**	**0.80**	**0.83**
Mainstream	**0.80**	**0.85**	**0.83**

Table 7: Results of 5-fold cross-validation of test data

	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Accuracy	0.83673469	0.80737705	0.84016393	0.83196721	0.83127572

[15] **Accuracy** is the sum of true positives and true negatives divided by the total number of articles. **Precision** is defined as the number of true positives divided by the number of true positives plus the number of false positives. **Recall** is defined as the number of true positives divided by the number of true positives plus the number of false negatives. **F1** is defined as the harmonic mean of precision and recall. See https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score.

The accuracy rate of 0.826% is sub-optimal for a baseline model when compared to studies such as Jarmul 2017 (who achieved 0.893% using *tf-idf*). However, the author is confident that with added parameter optimization (n-gram parameters, POS inclusion/exclusion) a significantly higher accuracy rate closer to 90% could be achieved. The removal of noun phrases, for example, immediately boosted accuracy up to 0.857%, although this produced a stylistically uninteresting set of most-informative-features, made up mostly of verbs. Including noun phrases achieved a list of most-informative-features that was closer to the anticipated (and desired) lexical features the author envisaged from the outset of the study. These are discussed below.

Most-informative-features

The program made use of 4,263 features (word tokens) in total. The most-informative-features are those that are most significant in determining a class. In the case of the present classifier, *scikit-learn's* Naïve Bayes classifier works by making a default assumption about each text: that it is "mainstream". Each feature is then assigned a negative weighting that denotes the probability that the text is not "mainstream" ("hyperpartisan"). The greater the negative weight, the more informative the feature with respect to the "hyperpartisan" classification. The list below is thus a tabular representation of the 20 "heaviest" weights, alongside the 20 "lightest weights" – essentially, the results from opposite ends of the feature-weight spectrum. The 20 heaviest weights are assigned to the features that are most are most representative of "hyperpartisanship". Therefore, if a text featured the bigrams "black police" or "black lives" it was highly likely to be "hyperpartisan". Table 8 can also be understood as a list of "most hyperpartisan" and "least hyperpartisan" terms.

Table 8: Most informative features. The left column is most indicative of "hyperpartisan" news, the right column of "mainstream" news.

-12.2911	be soley	**-2.7791**	be
-12.2911	black police	**-3.6885**	have
-12.2911	document have	**-3.9336**	say
-12.2911	kneel	**-4.1610**	not
-12.2911	leftist	**4.4767**	do
-12.2911	liberal media	**-5.0937**	debate
-12.2911	medium be	**-5.1148**	campaign
-12.2911	move here	**-5.1820**	do not
-12.2911	rob	**-5.2541**	will
-12.2911	slaughter	**-5.2558**	would
-12.2911	view express	**-5.4273**	go
-11.5980	be truly	**-5.4656**	make
-11.5980	be upset	**-5.4854**	more
-11.5980	black lives	**-5.4965**	people
-11.5980	blatant	**-5.5377**	there
-11.5980	can watch	**5.5494**	when
-11.5980	disturbing	**-5.5990**	also
-11.5980	everyday	**-5.6460**	be not
-11.5980	evidently	**-5.6512**	state
-11.5980	furthermore	**-5.7174**	get

Right-wing bias

The author's intuition from the occurrence of 1) *leftist,* 2) *liberal media,* 3) *kneel*[16] and 4) *black police* as heavily weighted "hyperpartisan" terms, was that they were indicative of a right-wing bias and, potentially, politically incorrect language, polemical language, defamation/hate speech and conspiracy theories.

This is borne out by a concordance search for these terms using *SketchEngine*. The following are examples of these terms in context from the "hyperpartisan" dataset:

1) "***Leftist*** *loon Chris Hayes, host of little-watched All In With Chris Hayes on MSNBC, took to Twitter to bless us with his idiocy on Monday* […]".

(defamatory and insulting language)

[16] 22 out of 27 occurrences of the word "kneel" in the "hyperpartisan" corpus are in reference to the act of kneeling during the U.S. national anthem. This gesture was popularized by the American footballer Colin Kaepernick who performed this act as a protest against police violence toward African Americans. It has since become a point of contention in US politics. See for example Mindock (2019).

2) "*The Clinton political machine that used to control the message through the **liberal media** in the 1990s simply can't hide from today's alternative media*".

(conspiracy theory)

3) "*Ever since Colin Kaepernick started this cop-hating gesture by sitting and then **kneeling** during the National Anthem, it has had me furious*".

(polemical language)

4) "*After family members stoked angry crowds by saying that Keith Lamont Scott was holding a book when he was shot, and not a gun as the **black police** chief explained, mobs of black youth began looting, rioting, and attacking police & civilians*".

(politically incorrect language, polemical language)

Overall, the most informative feature results build on the results and intuitions of Reid (2019) by demonstrating that "hyperpartisan" style indicators correlate with polemical language in addition to political incorrectness, conspiracy theories and defamatory language. The author proposes the "hyperpartisan" class as a useful "alarm" for problematic content.

3.2.7.2. Problems

3.2.7.2.1. GENERALIZATION

So far, the model has provided useful qualitative and quantitative insights. But the author also wished to test whether the program could deal with completely exogenous data from outside the *BuzzFeed-Webis* corpus. This would be another test of the model's *generalization* capacity. Generalization in a machine learning context means a model's ability to perform well on new data (Müller, Guido 2016, p17). *Overfitting* occurs when the model learns very specific features of a dataset to the extent that it performs very well on training data, but then struggles with test data. This did not happen in the case of the Naïve Bayes classifier. However, an extraneous dataset from additional news websites would be another, more stringent test. This proved to be the case.

The author conducted a small-scale test using four news articles, two sourced from right-wing outlets, two from centre-left sources (see Table 9 below). The size of the texts corresponded roughly to that of the *BuzzFeed-Webis* dataset (between 1,100 and 1,300 words).

Table 9: Exogenous news articles fed into the machine learning pipeline (incorrect classes in red)

Title	Source (online)	Political bias	N-Bayes classification
Germans Under Siege: Police stats reveal Germans overwhelmingly victims of migrant crim	**FrontPage Mag**	Right/far-right	Hyperpartisan
https://www.frontpagemag.com/fpm/274676/germans-under-siege-stephen-brown			
Ginger Jihadis: Why are reheads attracted to radical Islam	**Breitbart News**	Right/far-right	Mainstream
https://www.breitbart.com/europe/2014/09/09/ginger-jihadis-why-redheads-are-attracted-to-radical-islam/			
Carrier deal saves Indiana jobs, but Trump critics fear dangerous preceder	**Guardian US**	Centre-left	Mainstream
https://www.theguardian.com/us-news/2016/dec/01/donald-trump-carrier-deal-manufacturing-jobs-mexico			
They Died Shielding Their Baby in El Paso	**New York Times**	Centre-left	Hyperpartisan
https://www.nytimes.com/2019/08/12/us/el-paso-anchondo-jordan-andre.html			

Although this is a very small-scale test, it is not encouraging to see that two out of four texts were classified incorrectly.

This raises the question as to whether the model is generalizable enough. Generalization and the problem of overfitting (and underfitting) are closely linked to the size of the dataset. Larger datasets can be used in conjunction with more complex models without overfitting (Müller, Guido, 2016 p. 29). The model employed by the author was simple *and* used a small dataset (887,943 words and 1,627 texts), which is in keeping with the recommendations in the literature. To improve performance and generalization, the model needs more data and thus more complexity to accommodate new features. These new features would be the result of different writing styles, which is to say, new words and word patterns. A follow up study might incorporate multiple datasets on a much larger scale.

3.2.7.2.2. CIRCUMVENTION

It is not difficult for a committed creator or disseminator of mis-/disinformation to circumvent a style-based detection tool. This could be achieved either through meticulous human labour or through Natural Language Generation tools. Both of these approaches are capable of producing an article with "*Breitbart* content" in the "style of the *Guardian*" (*Interview 1*, see Appendix), where convincing style and tone lend credibility to misleading information.

As a solution, the author suggests augmenting a style-based tool with the fact-checking and context-based methods discussed in Section 2.3.

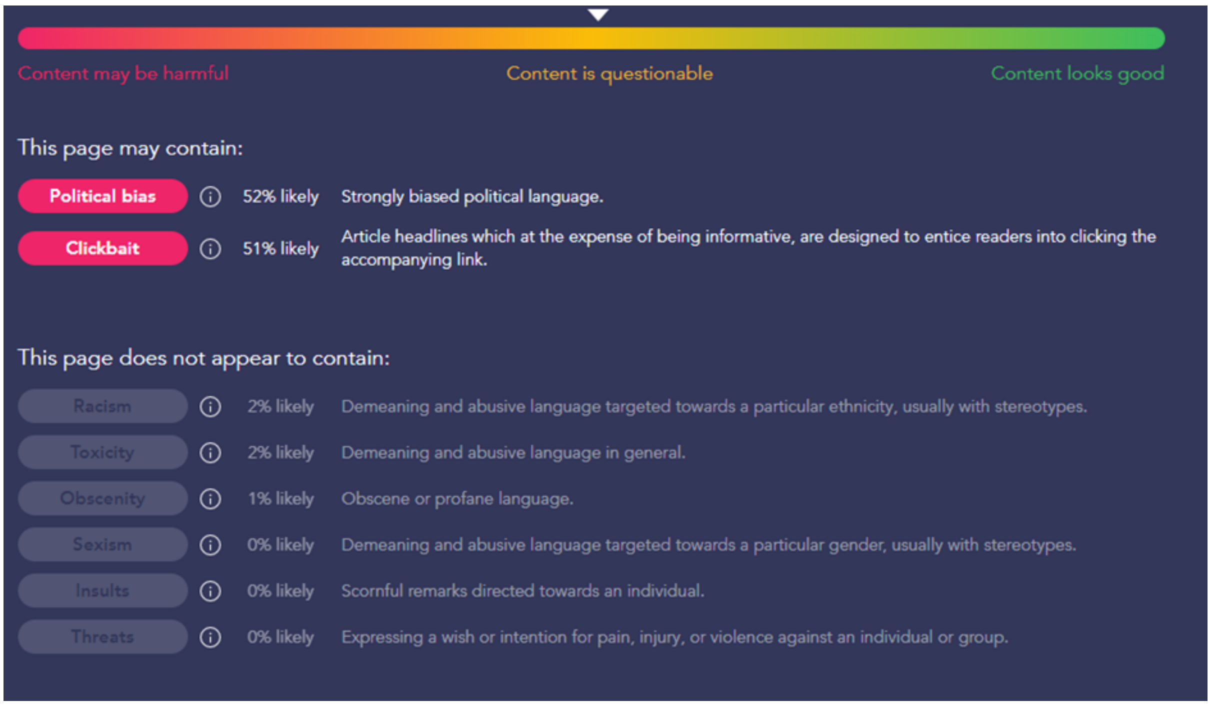
3.2.8. Integration suggestions

In terms of integration into user-friendly software solutions, a front-end user-interface was beyond the scope of this study. The optimal graphical interface from a user experience perspective would be a browser-based application that allows the user to paste a URL into a search box. This would then trigger the back-end process coded for the purposes of Case Study 1 and output a classification.

Although the “mainstream”/”hyperpartisan” binary is useful, in that its scope of interpretation is broad, a more nuanced classification would be desirable in every case.

In design terms, and particularly with respect to ease-of-use, visual clarity, and nuance of output, the likelihood-based classification tool FACTMATA[17] is a prime example of an excellent interface. The user simply copies and pastes a URL into a text box and then receives a multi-tiered evaluation with eight content-based metrics. The output below shows the likelihood that the *Breitbart* article “Ginger jihadis: Why redheads are attracted to radical Islam” is “questionable” (with respect to veracity).

Fig. 3: Results from *FACTMATA*



This page may contain:

Political bias	52% likely	Strongly biased political language.
Clickbait	51% likely	Article headlines which at the expense of being informative, are designed to entice readers into clicking the accompanying link.

This page does not appear to contain:

Racism	2% likely	Demeaning and abusive language targeted towards a particular ethnicity, usually with stereotypes.
Toxicity	2% likely	Demeaning and abusive language in general.
Obscenity	1% likely	Obscene or profane language.
Sexism	0% likely	Demeaning and abusive language targeted towards a particular gender, usually with stereotypes.
Insults	0% likely	Scornful remarks directed towards an individual.
Threats	0% likely	Expressing a wish or intention for pain, injury, or violence against an individual or group.

[17] https://try.factmata.com/

3.2.9. Checklist for Case Study 1

UTILITY CHECKLIST: CASE STUDY 1

- ✓ Necessity – Local: directly tackles partisanship in news media
- ✓ Trust – Transparent: thanks to most informative features
  Efficient: 82.55% accuracy can be improved, Naïve Bayes is fast
- ☒ Practicality – User-friendly: front-end application required
  Nuanced output: more nuanced gradation necessary

Case Study 1 sought to examine how close a simple style-based naive baseline document classifier could come to fulfilling the utility criteria set out in Section 1 of this study.

With respect to *local necessity*, the program is conceived as the programmatic basis for a web-based interface – primarily as a browser application that accepts an input (URL) and then classifies the content contained therein. Such a program could also conceivably function as a plug-in that automatically flags the content of a webpage as soon as it is accessed. These would serve as useful early warning signs for news consumers who wish to stay alert to the mis-/disinformation threat.

In terms of *trust*, the most informative features function of the program provides *transparent* information on the words that trigger a "hyperpartisan" output. A "top 10 features per document" output would be a useful and transparent explanatory aid for an inquisitive or aggrieved data subject. The *accuracy* statistics demonstrate that the program is not quite efficient enough. However, significant improvements are possible by adjusting parameters. The tool also struggled with exogenous news articles. More data and additional complexity will improve generalisation. The model is *fast*, as is to be expected from a Naïve Bayes classifier (Müller and Guido 2016, p.68).

With respect to *practicality*, a web-interface is not part of the scope of the present dissertation, which means that *user-friendliness* cannot be evaluated. The output classes are concepts open to degrees of interpretation ("hyperpartisan" correlates closely to political incorrectness, conspiracy theory and polemic) but a *more nuanced* output with additional classes and metrics (in the style of FACTMATA) are desirable to provide more concise metrics.

Ultimately the tool is capable of fulfilling the promise of Rubin et al's (2015) assertion that a " text analytical system" can "enhance human abilities to spot lies, provide evidence for its suspicions, and alert users to further fact-checking." (p. 6)

This foundational model is a useful starting point for more complex architectural expansions incorporating features such as fact-checkers and meta-data analyses.