Predicting the formality of language using supervised machine learning

Jeffrey Waters

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

25th September 2020

Abstract

This project designs and implements a system for predicting formality classifications on a corpus of sentences. To do so, it employs sentence features: that is, data describing the characteristics of the sentences, such as each sentence's number of nouns. Those features were inspired by academic theories of formality and by metrics used to gauge the level of formality in documents.

Formality predictions are made using machine learning software. It uses classification and feature data to train itself to make formality classification predictions (using part of the dataset). It subsequently makes formality classification predictions on a different part of the dataset, using just the sentences' feature data. Each prediction can be compared with the corresponding sentence's actual formality classification, showing how effectively the test predicted formality classifications.

Little work has been done previously on predicting whether written communications are formal or informal. This project helps to address that knowledge gap. It tests features that have not previously been tested in formality prediction classification studies. For example, it explores how different types of n-gram compare in terms of their formality classification effectiveness. Moreover, it compares the formality prediction effectiveness of four major classifiers – that is, ways of representing and processing the data used by the machine learning software – which is something that has not been done previously.

As the level of formality in language reflects social dynamics (Pavlick & Tetreault, 2016), formality classification prediction may help machines to make sense of documents. This means that it has the potential to be used in a wide range of applications, including translating documents (Niu, Martindale, & Carpuat, 2017) and interpreting search engine queries. It is hoped that this project will assist those endeavours by contributing knowledge to this little studied area.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

Name: Jeffrey Waters Signature: *Jeffrey Waters*

Acknowledgements

I would like to thank my supervisor, Dr Graham McDonald, for the excellent guidance he has given me throughout the production of this project. I would also like to thank him for believing in me and reassuring me during times when I was experiencing self-doubt.

I would like to thank the project support team for the friendly, professional and efficient service they have provided. Everyone I dealt with was fantastic.

I would like to thank my mother for reading draft versions of this report and giving me valuable feedback. I would also like to thank her for the advice and encouragement she gave me throughout my MSc course (and throughout my life).

I would like to thank Mr Douglas MacFarlane, the School of Computing Science's Systems Manager, for setting me up with a virtual machine with which to run this project's tests, and for upgrading the virtual machine's memory so that I could progress through this project's machine learning tests far more quickly.

I would like to thank Dr Maxinne Connolly-Panagopoulos and Ms Michal Shimonovich of the University of Glasgow's Learning Enhancement and Academic Development Service for providing me with advice on matters relating to academic writing.

I would like to thank Dr Shibamouli Lahiri of the University of Michigan for making public the dataset that was used for this project's tests, and for responding to a query I emailed him.

Finally, I would like to thank my brother, Robert, who passed away recently. As someone with a PhD in Software Development, he was able to explain programming and general computing concepts to me, and spent many hours doing so throughout my MSc course. Robert was a great teacher. He went through advanced ideas using language that I was able to understand, and when he was explaining something, he ensured that I properly understood the material before moving on. In addition, he was always happy to provide me with general advice and encouragement relating to the course, and I could count on him to be open and honest with me. Thank you, Robert. Your support was invaluable.

Contents

Chapter 1 - Introduction

This chapter begins by discussing this project's areas of investigation. It then discusses its aims, why the matters investigated by this project are worthy of investigation and how this project contributes to existing knowledge. Finally, outlines the structure of the report.

1.1 Areas of investigation

'You are cordially invited to attend.' 'Hey, why don't you come along?'

The two sentences above communicate a similar message, but most people would probably agree that the first one is expressed in a formal manner and the latter in an informal manner. This project engineers a system to predict by automated means whether written sentences have been classified as formal or informal. A machine learning program is given sentence features for each sentence – that is, data describing sentence characteristics, such as the number of verbs – and is informed whether each sentence is formal or informal (based on ratings provided by human raters). The program models the relationship between the feature data and the formality classifications using part of the data, and predicts whether each of the sentences in the remaining part of the data is formal or informal. These predictions can be compared against the actual formality classifications, which allows the effectiveness of the predictions to be assessed. Through running many such tests, a picture emerges of which features appear to work best when carrying out formality classification prediction tests.

1.2 Project aims

This project aims to provide insights into formality classification prediction as applied to written communication. It runs machine learning tests on a corpus of sentences, using a variety of features. Those features include various types of n-grams, with and without stop words, and with every data representation (these terms are explained in Section 2.3). As well as testing n-grams, it tests other features, which fall into four categories: punctuation, lexical, sentiment and grammar. This project also compares the effectiveness of four major classifiers, which are ways in which the feature and classification data is modelled and processed by the machine learning program.

Some questions about the test results are posed in Section 5.1 and answered in Section 6.4. It is hoped that the answers to the questions and the results themselves will provide insights into formality classification prediction of written communication, and contribute to a little-studied area of computing with wide-ranging potential applications (see Section 1.3 below).

1.3 Why is formality classification prediction of written sentences a worthwhile area of investigation?

Formality and informality reflect underlying social dynamics, such as how familiar people are with each other (Pavlick & Tetreault, 2016). As a result, formality classification prediction studies of written communication can illuminate how other social dynamics are expressed in words, grammar and punctuation. Moreover, they bridge the gap between merely having theories of how formality in written communication is manifested and being able to systemtically predict whether a document is formal or informal.

Formality classification prediction can potentially help in situations where an assessment of the social dynamics underlying the communication is useful, such as translating documents (Niu, Martindale, & Carpuat, 2017) and interpreting search engine queries. It might also help to generate accurate summaries of documents by automated means (Chawla, et al., 2019). In addition, it might also help to adapt autoprompt suggestions to the writer's usual writing style. Another potential application is helping to identify clickbait (Daoud & El-Seoud, 2014).

1.4 How does this project enhance existing research?

Much has previously been written about formality, but there have been only a handful of studies that have focused on predicting formality classifications. For example, the F-Score is a widely used metric of formality (Lahiri, 2015). However, the fact that it can be used to gauge document formality does not mean that it can be used successfully to predict formality classifications derived from human ratings.

The fact that so little work has been done in the area of document formality classification prediction means that this project was able to study aspects of it that have not been studied previously. In particular, to the best of the author's knowledge, there have been no previous formality classification prediction studies which have compared the predictive effectiveness of: unigrams, bigrams and trigrams, binary, non-binary and TF-IDF representations, n-grams with and without stop words included[1], and all the four classifiers that are used. By studying these areas, this project will help to fill knowledge gaps within the field of formality classification prediction.

1.5 Report structure

Chapter 2 discusses academic theories of formality and formality metrics. It then introduces one of the project's features, n-grams, before explaining how the formality theories and metrics informed the decision to employ n-grams and the other features used in the tests.

Chapter 3 presents an overview of the system architecture requirements for running a formality classification study on a corpus of documents.

Chapter 4 provides the rationale of some of the major design decisions. It then describes what the project's coding does and and how it works, and finally discusses the measures used to ensure that the code functions properly and that the data is free of duplicate entries.

Chapter 5 lists eight questions that the project seeks to answer. It then discusses the metrics used to assess the results of tests, and finally explains why the dataset that was used was selected.

Chapter 6 presents the results of the formality classification prediction tests and then answers Chapter 5's questions in light of the results.

Finally, Chapter 7 provides overall reflections on the project, as well as discussing the limitations of the tests and offering suggestions for future work.

[1] The terminology that is used up to this point in this sentence is explained in Section 2.3.

Chapter 2 – Background / related work

This chapter starts by exploring theories of formality, and then discusses metrics that have been used in studies to gauge document formality. It then discusses the features employed in the project and how their creation was informed by the formality theories and metrics.

2.1 Theories of formality

Graesser, et al. (2014) state that formal communication exhibits 'low narrativity, syntactic complexity, word abstractness, and high cohesion'. Heylighen and Dewaele (1999) take a similar view, stating that formality is characterised by 'detachment, accuracy, rigidity and heaviness'. They also take the view that formal communication is more efficient than informal communication.

Informal communication, by contrast, is 'more flexible, direct, implicit, and involved [but] more subjective, less accurate and less informative' according to Heylighen and Dewaele (1999). They give an example of informal language being more implicit: a person saying "It is cold here", when what they mean is "I would like the window to be closed". This way of communicating is referred to as implicature, 'the act of meaning or implying one thing by saying something else' (Davis, 2019). The view that informal language is more implicit and less accurate than formal language is supported by the observation by Lahiri, et al. (2011) that informality is characterised by a greater degree of non-explicit, context dependent language than formality. For example, in 'I will do it later', the word 'later' is an example of this phenonemon, as its precise meaning can only be known if one knows when the utterance occurred.

In addition to the above examples of informal communication, Li, et al. (2015) observe that informal communication contains what they refer to as 'informal lexical embedding' (they give 'hi there' as an example). Pavalanathan, et al. (2017) provide further examples: 'hedges (e.g., might, kind of), discourse markers (e.g., actually, I mean), and backchannels (e.g., yep, um)'.

Abu Sheikha & Inkpen (2011) provide some additional views on the differences between formal and informal communication. They believe that formal writing is impersonal, whereas informal writing is personal, and they state that informal writing uses 'complex words and sentences to express complex points' (whereas informal writing uses 'short, simple words and sentences'). In addition, they associate the following with informal communication: use of the first and second person rather than the third person, Anglo-Saxon derived terms (as opposed to ones of Latin origin), slang, abbreviations, concatenations (such as can't), and familiar salutations.

For a machine learning formality classification prediction study, Peterson, et al. (2011) hypothesised the following were signs than an email was informal: exclamation marks, sentences without terminating punctuation, ellipses, sentences entirely in lower case, and sentences whose first word was in lower case. The results of their study suggest that these linguistic features are effective for the purposes of formality classification prediction.

2.2 Metrics used to measure formality in previous studies

The F-Score is a formality metric: (noun frequency + adjective frequency + preposition frequency + article frequency – pronoun frequency – verb frequency – adverb frequency – interjection frequency + 100) / 2. Heylighen and Dewaele (1999) explain that the F-Score is based on the premise that words whose meaning is not context-dependent are more frequently found in formal communication, and that some types of words are context-depedendent and others not context dependent.

The Flesch-Kincaid grade level is a measure of document readability. Higher Flesch-Kincaid scores are believed to result in greater readability, and are associated with shorter sentences and fewer syllables per word (DuBay, 2006). Pavlick & Tetreault (2016) found in their study that higher Flesch-Kincaid scores had a positive relationship with higher levels of formality. Similarly, Mosquera & Moreda (2012) use the RIX readability measure to assess document formality. The formula for RIX is LW/S, where 'LW' is the number of words with more than 7 letters and 'S' is the number of sentences. A higher score is corresponds to lower readability, meaning that words with more than 7 letters detract from readability according to this metric.

2.3 Introduction to n-grams

This section will introduce n-grams, which are one of the key features used in the project. It will first explain what they are, and then discuss what are referred to as stop words.

2.3.1 What are n-grams?

N-grams are combinations of contiguous words within a document, going from left to right through the document, each of 'n' words in length. Take the following sentence: 'The cat sat on the mat'. With unigrams (n=1), each word in the sentence would become a part of an n-gram. With bigrams (n=2), the n-gram would be comprised of: 'The cat', 'cat sat', 'sat on', 'on the' and 'the mat'. With trigrams, the n-gram would be comprised of: 'The cat sat', 'cat sat on', 'sat on the' and 'on the mat'. Similarly, a 1,2 gram employs unigrams and bigrams, and a 1,2,3 gram uses unigrams, bigrams and trigrams.

2.3.2 What are n-gram data representations?

The n-gram related information about each sentence is conveyed to the machine learning program using one of three representations: binary, non-binary and Term Frequency-Inverse Document Frequency (TF-IDF). With binary n-grams, the machine learning algorithm is informed of the presence or absence in each sentence of every n-gram instance in the corpus, but not of their frequency of occurrence. With non-binary representation, the machine learning program is told how frequently every n-gram instance within the corpus occurs within each sentence. Finally, TF-IDF provides a weighted measure of both how frequently each n-gram instance occurs within the sentence under consideration and within the corpus of sentences.

To illustrate the previous paragraph, consider a unigram test where the corpus contains the word 'cat', which appears twice in a particular sentence. With binary representation, this is represented by '1' for that sentence, indicating the word's presence in the sentence. With non-binary representation, it is represented for that sentence by '2'. With TF-IDF, a figure is

used which corresponds to a balance of how frequently 'cat' appears in the sentence and how frequently it appears in the corpus of sentences.

As well as testing the classification prediction effectiveness of n-grams, this project explores how binary, non-binary and TF-IDF representations compare at formality classification prediction.

2.3.3 What are n-gram stop words?

When running n-gram based tests, it is possible to remove from consideration what are referred to as stop words. Stop words are extremely common words which do not add much meaning to a sentence, such as 'the', 'a', 'do', 'be', 'will', 'on', 'around', and 'beneath' (Hackeling, 2014). One of the things this project explores is how stop words affect formality classification prediction effectiveness.

2.4 Why n-gram only tests were employed in this project

The previous two sections provide a strong basis for believing that there are significant differences between formal and informal communication. It is therefore likely that certain words and phrases are more associated with formal or with informal communication. Take, for example, Abu Sheikha & Inkpen (2011)'s observation that 'I' and 'you' are particularly prevalent in informal communication. If most sentences containing 'I' and 'you' in the dataset are classed as 'informal', then through the use of unigrams the machine learning program would make a connection between the inclusion of those words and the 'informal' classifcation label. It would take that into consideration when reaching its formality classification prediction.

Certain two or three word phrases may usually be associated with either formal or informal communication. Through the use of bigrams and trigrams respectively, the machine learning program can take the associations into consideration. For example, formal language may be more likely to contain linking phrases like 'it follows that' and 'as a result' than informal language, as it is more syntatically more complex. Conversely, there exist informal, colloquial phrases, such as 'hi there' and 'kind of'.

2.5 Why the classification effectiveness of stop words was tested

N-gram tests are implemented both with stop words and with stop words excluded, so a formality classification prediction effectiveness comparison can be made. Based on the F-Score, a case can be made for hypothesising that such words may predict formality and one can also be made that they may predict informality. Whatever their overall effect on formality classification, it seemed plausible that they help with formality classification prediction, so it was deemed a good idea to include them.

2.6 Why binary, non-binary and TF-IDF representation are compared

The extra information provided by the non-binary and TF-IDF n-gram representations (compared with binary representations) may improve prediction effectiveness. For example, with non-binary representation, if a word appears multiple times in a sentence, it may be a sign of repetitive, inefficient communication. Also, the TF-IDF figures for each word or

phrase partly reflect frequency of usage within the corpus (and therefore possibly within the English language), and it may be that knowing whether a word is frequently used can help with formality classification prediction.

2.7 Reasons for testing non n-gram features

Table 1 below presents the categories of non n-gram features, and the features employed for each category. The sections below will explore the rationale for including each feature.

Table 1 - Non n-gram features employed

Category	Feature names
Punctuation	Number of capitalised words, Number of commas, Number of exclamation marks, Number of full stops, Number of question marks.
Sentiment	Implicature, Informativeness, VADER sentiment score
Grammar	F-Score, Number of adjectives, Number of adverbs, Number of conjunctions, Number of determiners, Number of existential theres, Number of interjections, Number of modal verbs, Number of nouns, Number of prepositions, Number of pronouns, Number of proper nouns, Number of verbs
Lexical	Average word frequency, Number of stop words, Number of words in 35 most common words in corpus, Number of words with < 5 characters, Number of words with > 7 characters

2.7.1 Reasons for punctuation-related features

The use of commas might be a sign that a document is formal, because they are evidence of syntactical complexity. Conversely, exclamation marks are possibly more likely to be found in informal communication, given that formal communication is associated with detachment and heaviness. Finally, the absence of a full stop or the presence of multiple full stops, question marks or exclamation marks suggests that the sentence may have been punctuated incorrectly, which is something that could be a sign of informal communication.

2.7.2 Reasons for sentiment-related features

If a communication is expressed in a state of high emotion such as excitement or anger, it is more likely to be informal. The is why VADER sentiment score was chosen as a feature. It produces a score of between -1 and 1, where -1 is extreme negative sentiment and 1 is extreme positive sentiment (Hutto & Gilbert, 2015).

Implicature was included as a feature because, as previously discussed, it may be linked with informal communication.

Finally, informativeness was used as a feature because it relates to clear and direct communication, which is associated with formality (Lahiri, 2015).

2.7.3 Reasons for grammar-related features

For the grammar category, the reasons for the feature choices are shown in Table 2 below.

Table 2 - Reasons for including each of the grammar-related features.

Grammar feature	Rationale for inclusion
Number of existential theres and number of modal verbs.	The former involves variations of 'there is' or 'there are'. This was thought most likely to happen with formal communication, due to its abstract nature. Modal verbs are words like 'could', 'may' and 'might'. They have been included because they involve abstract thinking and could therefore be more associated with formality.
Number of proper nouns	These are names of people, places and organisations. They were thought more likely to feature in informal communication, due to its narrative quality.
Number of conjunctions	These are connecting words, such as 'and', 'therefore' and 'despite'. It is hypothesised that they would be most likely to feature in formal sentences, due to their syntactically more complex nature.
Number of adjectives, number of adverbs, Number of interjections, Number of nouns, Number of prepositions, Number of pronouns, Number of verbs	These are like the components of the F-Score (except that the F-Score uses the frequency of various word types, rather than the outright number of them). It was therefore felt that there was a good basis for believing them to be effective formality prediction classification features.
F-Score	This metric was used as a feature to test if it can predict of human rating derived formality classifications, as well as being a measure of formality.

2.7.4 Reasons for lexical-related features

The inclusion of the average number of syllables per word was inspired by the connection between readability and formality. Texts with fewer syllables may be easier to read and therefore more likely to be formal.

The metrics relating to word length were chosen because they could be effective formality classification prediction features, even though it is not clear what their effect on formality classification is likely to be. On the one hand, longer words are likely to have more syllables than shorter words, meaning that readability is harder. Moreover, the previously discussed RIX readability measure indicates that having words greater than seven characters in length detracts from readability. On the other hand, abstract words and words of Latin origin seem generally to be longer than more everyday words and words of Anglo-Saxon origin and are associated with formality.

'Number of stop words' was included because, as discussed in Section 2.5, it was thought that stop words may be effective at predicting formality. Similarly, 'Number of words in 35 most common words in corpus' was included because it was hypothesised that the most common words within the corpus are commonly used words within the English language.

Chapter 3 - System architecture requirements

This chapter outlines the tools that are required to carry out a set of formality classification prediction tests on a corpus of documents, in terms of the dataset and the software. It also describes how the components interact (and depicts this arrangement in Figure 1).

3.1 Dataset requirements

A dataset of documents is needed, the requirements of which are given in Table 3 below.

Table 3- Dataset requirements

Requirement	Why it is needed
Formality classifications (or data to allow them to be derived) are needed.	There needs to be a way of determining each document's formality classification.
The data needs to be stored in a format such it is clear which data belongs to which sentence.	This is needed to make sense of the data.
The sentences need to be unique.	This prevents the same sentence being considered more than once.
The dataset needs to be sufficiently large for meaningful conclusions to be drawn.	If the dataset is insufficiently large (say, just 10 records), the sample size is insufficient to draw meaningful conclusions.

3.2 Software requirements

Software is also needed that will be able to implement the functionality described in the two sections below. For simplicity, a distinction is made between non-machine learning software and machine learning software, and each is discussed separately.

3.2.1 Non-machine learning software

The non-machine learning software begins by uploading the data. It then processes the data to provide the machine learning software with the following for each document: the document's formality classification, the feature(s) of the document to be tested (such as the number of adjectives), and which classifier to use. In the case of tests involving n-grams, it will also tell the machine learning software whether to include stop words and whether to use a binary, non-binary or TF-IDF representation. Finally, when the test has been conducted, the non-machine learning software is provided details of the predictions made by the machine learning software. It can then test each prediction against the corresponding actual classification and produce statistics summarising the test results.

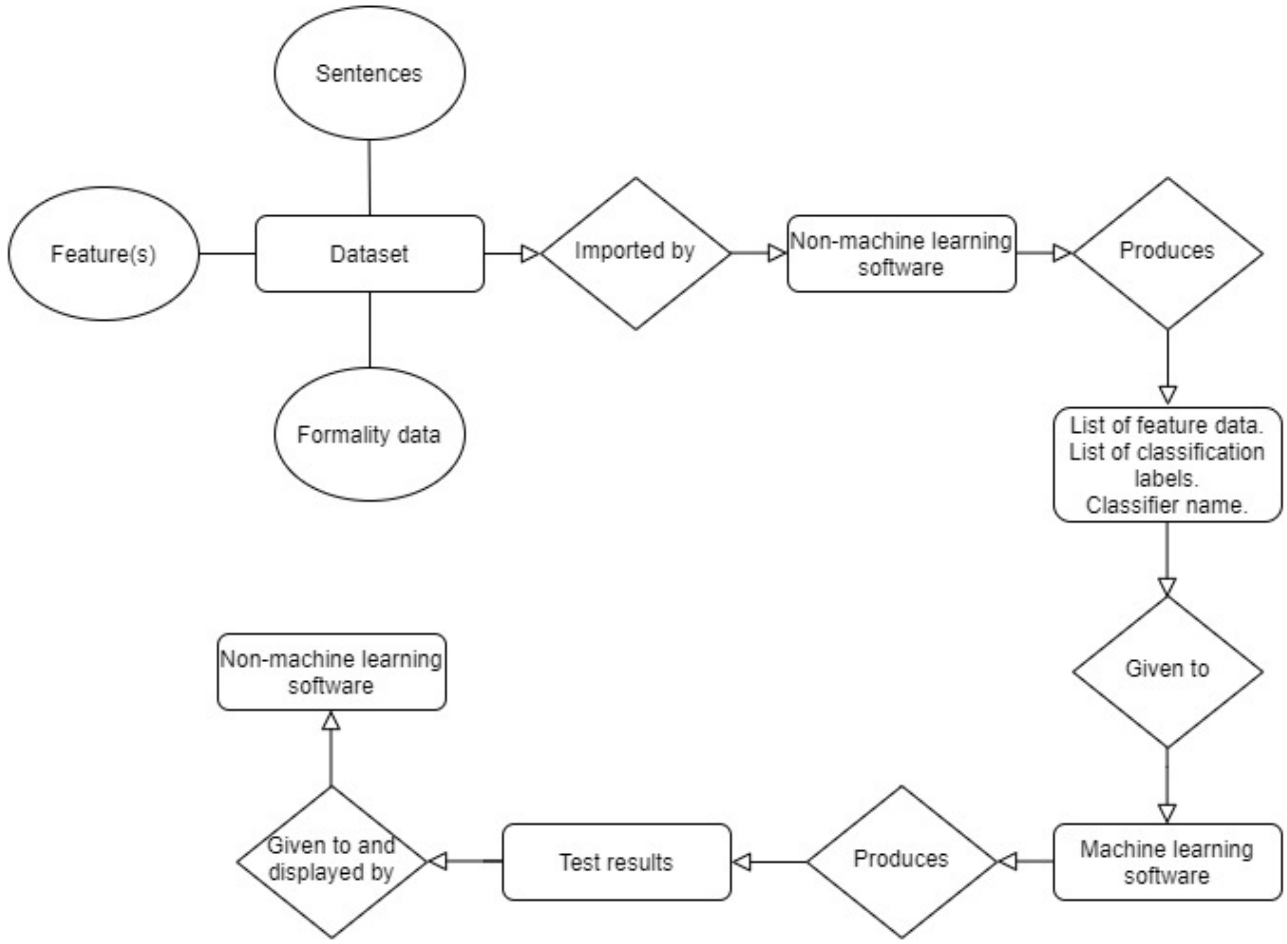
In addition, non-machine learning software will need to be able to produce additional metrics from the existing data, if required.

3.2.2 Machine learning software

The machine learning software is given the feature and formality classification data of each sentence by the non-machine learning software. The machine learning software implements training on part of the dataset – that is, learning to predict formality classifications – and then carries out testing on the remainder of the dataset (that is, making formality classification predictions). It then provides its predictions to the non-machine learning

software, so that the predictions can be compared against the actual classifications, and performance statistics generated.

Figure 1 – Entity relationship diagram illustrating the sequence of events for running machine learning tests



Chapter 4 – Implementation

This chapter begins by discussing the rationale behind some of the major test decisions. It then discusses what the project's code does and how it works. Finally, it describes the measures taken to ensure that the code functions properly and that the dataset is free of duplicate entries.

The code, test-related material and sample console output for the machine learning tests have been placed in the files submitted alongside this report, as well as on GitHub: https://github.com/JeffW12345/formality-classification.

4.1 Reasons for using supervised machine learning

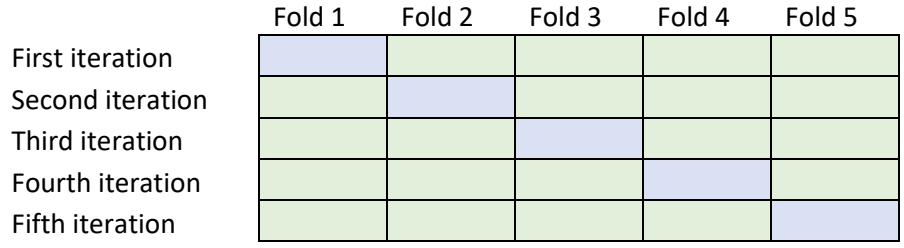
With supervised machine learning, a dataset consisting of features and classification labels are given to the programme. Predictions are made using that data, and statistics relating to those predictions are generated. That makes it ideal for investigating formality classification prediction.

4.2 Reasons for choosing scikit-learn and Python

The machine learning software that was employed, scikit-learn, meets the criteria required of machine learning software described in Section 3.2.2. In addition, it is highly respected. (VanderPlas, 2016).

A further benefit of using scikit-learn is it supports K-Fold cross-validation. With this approach, the dataset is split into 'k' folds. Folds are parts of the data put aside for a particular purpose. In the cases of this project's tests, the value of k was five. As a result, for this project's tests, the program would run five sets of tests, each using four fifths of the data for training purposes and a fifth for testing purposes, and using a different fifth for training purposes each time, as shown in Figure 2 below.

Figure 2- K-Fold cross-validation where k = 5. The blue represents a fold reserved for testing, and the green represents folds used for training.



Once all five tests have been completed, the machine learning program presents the user with the most successful set of results. The advantage of this approach is that, after each of the first four tests, the settings used can be tweaked to improve the performance of the remaining tests. As scikit-learn can randomise the order of the records within the dataset when K-Fold cross validation is used, any sequential bias in the dataset can be removed.

Scikit-learn works in conjunction with the Python programming language, so Python was the natural choice for all the programming requirements.

4.3 Reasons for the formality classification labelling approach

In the dataset, there are formality scores ranging between 1 and 7 for each sentence, which had been provided by human raters. The reason for using ratings provided by humans as opposed to using an algorithm-derived formality metric (such as the F-Score) is because it was felt that human judgement would capture the social dynamics suggested by a sentence in a way that could not be achieved using an algorithm.

A decision was made to label formality scores >= 4 as formal, and scores < 4 as informal, as 4 is the halfway point on the scale.

4.4 Reasons for the classifiers chosen

The classifiers used are Logistic Regression, Multinomial Bayes, Random Forest and Support Vector Machine. These classifiers used were chosen partly because they are well respected, tried-and-tested classifiers. They were also selected because they operate in different ways, and, as a result, they may differ in terms of their effectiveness at formality classification prediction. A technical discussion of how the classifiers work is beyond the scope of this paper, but Brownlee (2020, August 14) provides an excellent discussion of how classifiers differ in operation.

4.5 Code details

Several modules were used for this project, each containing different functionality. This separation of responsibilities makes the code easier for the coder to navigate, and therefore to maintain and extend. It also means that the user can run the module that is relevant to their requirements, rather than having to choose between a list of options.

With all the modules, the user is not required to modify the code. Instead, they simply execute the relevant Python module, and are prompted to enter their requirements before being presented with output that corresponds to that information. This approach was done to result in a better user experience and to avoid the user accidentally damaging the code.

For some of the functionality, code taken the internet sources was used. When this happens, the source is acknowledged in the comments within the code.

The major functions of the code are discussed in the subsections below.

4.5.1 Producing McNemar stats

The code for producing the McNemar co-efficient values (discussed in Section 5.2), mcnemar-stats.py, prompts the user to ask for the number of true positives, false positives, true negatives and false negatives for each of two sets of results. It then gives the user the McNemar coefficient and the associated p-value, and states whether the difference in classification prediction effectiveness between the two results is statistically significant to a p value of 5%.

4.5.2 Creating new features and writing them to the data file

A module called add-new-data is used for this functionality. It uses a function called loadData() to import the data from a comma-delimited CSV file. The loadData() function starts by using a helper function to ask the user to confirm whether the default data file is

the file being used, and to enter the correct file name if not. This is done because the user may have renamed the dataset or may wish to use a different dataset to the one used for this project (for example, because they may plan to run formality classification prediction tests on that dataset). It then uses another helper function to check if the data file is present. If the data file is not present, the user is informed, and the program exits.

Next, loadData() loads the data in the top line of the data file – which contains the field headers, as shown in Figure 3. It subsequently works through the data line by line, until it has processed all the data in the file.

Figure 3 - Excerpt from the original data file (viewed in Excel). The field names are on the top row, and each row below that row contains a separate sentence and the data relating to it.

Sentence	HIT ID	Formality	Informati	Implicatur	Length in	Length in	F-score	I-score	Lexical De	Actual sentence
0	3D1TUISJV	5.8	6.4	4.2	33	201	93.93939	7.242424	66.66667	10In High Bay 4 of t
1	3GKAWYF	5.2	5.8	2.6	45	266	85.55556	5.2	62.22222	12The oxygen vent
2	3VMHWJB	5.2	5.6	4.8	29	186	93.10345	3.896552	65.51724	13In the Rotation

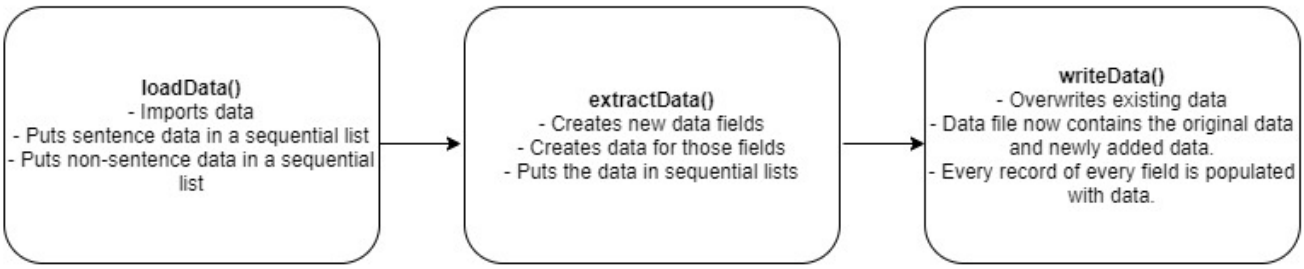
The loadData() function creates two lists, whose entries are stored in the order in which they are uploaded from the data file. One is a list of data relating to the sentences (such as formality scores), called 'dataExcludingSentence'. The other is a list of the sentences themselves (called 'corpus'). Some of the sentences contain commas, so using two lists allows those commas to be preserved. These lists are in the same order, making it possible to determine which data relates to which sentence. The 'dataExcludingSentence' list is a list of lists: for each sentence, there is a list of data items relating to that sentence.

The extractData() function creates from the existing data new data fields, to be used as features. It runs a 'for' loop through the corpus of sentences, extracting new data from the sentences and putting them into lists, which each list relating to a separate field. These lists are ordered in the same order as the list of sentences, so it is possible to determine which data relates to which sentence. It also relies on outside libraries - the NLTK library is used to determine each word's word type (noun, verb, etc) and also to count the number of syllables in each word, and an outside library was used to calculate the VADER sentiment score.

The writeData() function writes data back to the file, overwriting all the previous data. The new data fields are placed between the end of the original non-sentence fields and the sentence field. The reason for making the sentence the final field is because some sentences contain commas, and those commas would be indistinguishable from field delimiters otherwise.

In summary, existing data is loaded into the program, new fields are created and populated using that data and using outside libraries, and then both the original data and the new data are written to the original data file. The non-helper functions within add-new-fields.py are summarised in Figure 4 overleaf.

Figure 4– Summary of the non-helper functions used by add-new-fields.py. The arrows show the running order.



4.5.3 Running machine learning tests

The ngram-tests-only.py module is for running tests with n-grams but no other features, non-ngram-tests-only.py is for tests which involve purely features other than n-grams, and ngram-and-non-ngram-tests-combined.py is for tests which combine n-grams and other features into a single test.

The loadData() function of for all three modules is identical, and it imports data in the same way as its counterpart in add-new-fields.py does. In addition, it assigns each record a formality classification when that record is uploaded, which it calculates using the formality score field. The formality classifications are put in a list of Boolean values, and the formality classification of a sentence is 'true' if the formality score is >= 4, and false otherwise.

What happens next differs between the modules. With ngram-only-tests.py, the user is asked a series of multiple choice questions to ascertain what the parameters for their test are in terms the n-gram type, the n-gram representation (binary, non-binary or TF-IDF), whether stop words are included, and the classifier. An example multiple choice question is presented in bold below:

The classifiers are:

1 - Support Vector Machine
2 - Logistic Regression
3 - Multinomial Bayes
4 - Random Forest

Please choose a classifier by typing a number between 1 and 4 and then press 'enter':

With non-ngram-tests-only.py, the user is presented with numbered available features. They are invited to select a feature to test by entering the relevant number then 'enter'. Once they have done so, they are invited to select another feature to add (from a list which no longer includes the feature they have already selected) or to type 'C' then 'enter' to use purely the feature(s) already selected, at which point they proceed to the next stage (being asked to choose a classifier). Here is some example output:

So far, you have selected the following features:

Number of full stops
Number of question marks

You can add the following features:

1 - Informativeness
[Features 2 – 29 have been deleted from this example output.]
30 - VADER sentiment score

Please choose an additional feature and press 'enter'
or press C then 'enter' to select your classifier:

Finally, with ngram-and-non-ngram-tests-combined.py, the user is invited to submit all the information that they are asked to submit for the previous two modules.

With all the machine learning modules, if the user enters an invalid option, the function responsible for asking them to make a choice recursively calls itself, causing them to be asked the question again. In addition, with all the modules a human-readable summary of the test's features is generated, so that it can be included in the results summary.

Once the selections are made, the following are passed to a function called classificationResults: a vector containing sentence feature data, a vector containing the classification labels and the test description. The type of classifier is not passed, as that is stored in global variable. The classificationResults function informs scikit-learn that k-fold validation with randomisation will be used, and it instructs scikit-learn to run the tests.

Scikit-learn returns a list of predictions, which are stored in the same order as the list of classification labels, meaning that comparisons can be made between predicted and actual formality classifications. This allows 'classificationResults' to generate performance statistics (which are discussed in Section 5.2). This is an example of a results summary:

Feature(s) tested: 'VADER sentiment score' and 'Informativeness'

Classifier: Logistic Regression

Total predictions: 1406
TRUE POSITIVES: 519
FALSE POSITIVES: 203
TRUE NEGATIVES: 541
FALSE NEGATIVES: 143
Accuracy: 0.75
Precision: 0.72
Recall: 0.78
False positive rate: 0.27
AUC: 0.84
Balanced accuracy: 0.76

4.6 Testing the code

This section describes tests were carried out to check if the code produced the expected outcomes and whether the data was free from duplicate sentences. The files referred to in this section have been placed in the 'Program integrity test material' folder in the files submitted with this project and on GitHub.

4.6.1 Do fields for newly added features contain accurate data?

A dummy data sheet was created, Dummy_Data_Before.csv, with the same fields as are present in the original dataset, i.e. the one that existed before new features were added. All the data except the field headings were deleted. Twenty sentences created by an online random sentence generator were then added. In addition, random formality scores between 1 and 7 were then assigned to each record (as the human formality ratings are between 1 and 7), and remaining fields were populated with random numbers.

This code described in Section 4.5.2 wrote new, fully populated data fields with sentence-related data to the dummy file, which was subsequently renamed Dummy_Data_After.csv. Checks were then made to ensure that the data in the newly added fields was correct, by comparing it against the sentences (for example, checking if the number of nouns in each sentence corresponded with the number stated in the relevant field). The data in the newly created fields was as expected.

4.6.2 Do the machine learning tests produce accurate results?

Tests relating to the machine learning programs were run using Dummy_Data_After.csv to check they were operating as intended. Invalid answers were entered in response to user input questions to ensure that the program handled them correctly. In addition, the tests' feature and classification vectors were displayed in test prints, meaning they could be compared against the dataset. Also, for tests involving n-grams, a test print showing an internal representation of n-gram specifications was displayed, so that this information could be compared against the user's selections.

The tests were internally consistent and consistent with the dataset, showing that the machine learning programs operate correctly. Sample console output for the program integrity tests has been placed in a Word document, Program_integrity_tests.docx.

4.6.3 Were correct McNemar statistics obtained?

The user is prompted to enter performance data. A 'p value' is generated[2], which is checked against the 'p value' for the same data generated by a website[3]. The same 'p values' were obtained from both sources.

4.6.4 Did the dataset contain purely unique sentences?

Finally, a script was written to check whether the dataset used for the project's tests is free of duplicate sentences, as claimed by Lahiri (2015). The script, which is in the 'Program integrity test material' folder, confirmed each sentence within the corpus is unique.

[2] There is an explanation of what 'p values' are in Section 5.2.

[3] www.scistat.com/statisticaltests/mcnemar.php

Chapter 5 – Experimental setup

This chapter begins by listing the questions that the project seeks to answer (which are answered in Section 6.4). It then discusses the metrics that are used to assess formality classification prediction effectiveness. Finally, it discusses the reasons for selecting the dataset used in this project.

5.1 Questions to be addressed by the project

The project seeks to answer the following questions:

1. How effectively can n-grams predict formality classifications?
2. How do binary, non-binary n-and TF-IDF n-grams compare at formality classification prediction?
3. What effect do stop words have on formality classification prediction performance?
4. How effective are the features tested other than n-grams at formality classification prediction?
5. How effective is the F-Score at formality classification prediction?
6. How do the four categories of non n-gram features compare in terms of formality classification prediction effectiveness?
7. How do the four classifiers tested compare in terms of formality classification prediction effectiveness?
8. Does combining formality classification features result in better formality classification prediction performance?

5.2 Discussion of the metrics used

Each machine learning test can produce one of four possible outcomes for each document:

A. True positive (TP) – The sentence was predicted to be formal and was formal.
B. True negative (TN) – The sentence was predicted to be informal and was informal.
C. False positive (FP) – The sentence was predicted to be formal and was informal.
D. False negative (FN) – The sentence was predicted to be informal and was formal.

Various metrics can be derived from these outcomes, which together paint a picture of classification prediction performance. Most of these metrics are summarised in Table 4 below. A metric that is not covered in Table 4 is the area under the Receiver Operating Characteristic (ROC) curve, known commonly as the Area Under Curve (AUC). Its scores range from 0 to 1, and scores above 0.5 represent a classification performance than is better than would be expected from randomness. A benefit of using the AUC figure is that it is not distorted by the fact that there may be more formal than informal documents in the corpus, or vice versa (Hackeling, 2014).

The final metric not covered in Table 4 the McNemar coefficient. McNemar tests are used to determine what the likelihood is that the differences between the formality prediction classification effectiveness of two sets of results is due to chance. For example, it might test whether a particular feature outperformed the results that would be obtained from purely random predictions of whether sentences are formal or informal.

If the probability of the difference in classification between the two sets of results being due to chance, known as the p-value, is less than 5%, then the two sets of results are deemed to be significantly different for the purposes of this project. Otherwise, the two sets of results are deemed to be similar.

Table 4 - Description of some of the metrics employed in the project. TP = True positive, TN = True negative, FP = False positive, FN = False negative

Metric	**What it measures**	**Formula**
Accuracy	Proportion of predictions that were correct.	(TP+ TN)/(TP+TN+FP+FN)
Precision	Proportion of sentences that were predicted to be formal that are formal.	TP/(TP+FP)
Recall	Proportion of formal sentences that were detected.	TP/(TP+FN)
F-Measure	Balance of precision and recall.	(2 * Precision * Recall) / (Precision + Recall)
False Positive Rate	Informal sentences incorrectly predicted to be formal divided by the total number of informal sentences.	FP/(TN + FP)
Balanced accuracy	Balance of recall and false positive rate.	(Recall + False Positive Rate) / 2

5.3 Reasons for choosing the dataset

The dataset employed was used in a previous academic study into formality (Lahiri, 2015) and made public. There are several benefits of using this dataset, which was described by Lahiri (2015) as 'the largest sentence-level annotated corpus released for formality, informativeness, and implicature'. The dataset was chosen because of the benefits, which are summarised in Table 5 overleaf.

Table 5 – Benefits of choosing the dataset used for this project

Fact about dataset	**Benefit this provides**
The dataset contains a large sample of sentences (N = 7032).	A large dataset offers reduced variability. The benefit of low variability is that the results are less likely to be due to randomness. To illustrate variability, if a coin is tossed four times and comes up heads twice, that is not strong evidence that it is unbiased (as the variability is high), but it if is tossed 10,000 times and comes up heads 5,000 times, it is strong evidence of a lack of bias (as the variability is low).
The sentences are unique.	This characteristic prevents the results being distorted by repetitions of the same sentence.
Each sentence has a formality rating.	This enables formality classifications to be derived.
Each sentence in the dataset has a rating for informativeness and implicature	These fields can be used as features in tests.
Data is in CSV format, with the data for each sentence on a separate line.	Data stored in CSV format is readable by humans, allowing easy inspection of the data, and it is straightforward to handle.
The sentences are from three sources (blogs, from forums and from news articles).	This increases the diversity of sentences and may make them more reflective of sentences generally.
The sentences are from written communication, rather than being transcripts of speech.	This means that they are suitable for a formality classification study applied to written communication, which is this the type of communication that this project is studying.
The study had 527 formality raters (Lahiri, 2015).	This removes the possibility of an individual reviewer's personal biases significantly affecting the dataset.
The advertisement for formality raters asked for people from the US (Lahiri, 2015).	The annotators were likely to be proficient English speakers.
Each of the sentences was scored for formality by five reviewers.[4] The formality scores in the dataset represent the average score given.	The average of five estimates is likely to be more reliable than just one person's estimate, due to the 'wisdom of crowds' effect, whereby multiple perspectives are reflected in the average figure (Spiegelhalter, 2019).

[4] The source of this information is email correspondence between Dr Shibamouli Lahiri of the University of Michigan, who compiled the data, and this paper's author.

Chapter 6 – Results

This chapter begins by providing a description of the metrics used to assess the performance of the machine learning tests. It then presents the results of the tests, before using the results to answer the questions posed in Section 5.1 of the Experimental Setup chapter.

For all the tables in this chapter, the results that would have been obtained from random selections are given, as a benchmark comparison, and are presented on the bottom line of each table. The best performers for each metric in each table are in bold, for easy identification. If an item is in green, it means that it outperformed the random prediction benchmark in terms of classification prediction to a statistically significant degree according to McNemar tests (see Section 6.1 below). If it is in red, it did not.

6.1 N-gram results

McNemar tests show that all the n-gram results performed better than could be obtained by randomness, except for some of the trigrams.

Table 6 below compares the performance with stop words versus without stop words included in the tests. The remaining tables in this section, Tables 7-9, compare the classification prediction performances of the various types of n-grams, the n-gram representation types (TF-IDF, binary and non-binary) and the four classifiers.

Table 6 - Average AUC figures for stop words included vs stop words excluded. Best performing categories for 'stops included' and 'stops excluded' are in bold.

Category of data	Stop words included	Stop words excluded
Average for table (excluding benchmark)	0.81	0.77
TF-IDF	0.82	0.77
Binary	0.81	0.77
Non-binary	0.78	0.77
Unigrams	**0.87**	**0.86**
Bigrams	0.81	0.72
Trigrams	0.63	0.55
1,2 grams (unigram & bigram)	0.86	**0.86**
1,2,3 grams (unigram & bigram & trigram)	0.86	0.85
Support Vector Machine	0.84	0.83
Logistic Regression	0.83	0.78
Multinomial Bayes	0.79	0.67
Random Forest	0.78	0.79
Random prediction benchmark	*0.51*	*0.51*

The rest of the results in this section are for tests where stop words are used, and the figures represent the average of results across all the classifiers.

Table 7 - N-gram performance stats. Figures are averages. Stop words were used.

Type of n-gram	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F-Measure
Unigrams	**0.80**	**0.81**	**0.74**	0.15	**0.87**	**0.79**	**0.77**
Bigrams	0.72	0.80	0.56	0.13	0.81	0.72	0.65
Trigrams	0.59	0.79	0.21	**0.07**	0.63	0.57	0.28
Unigrams and bigrams	0.79	0.82	0.72	0.14	0.86	**0.79**	0.76
Unigrams, bigrams and trigrams	0.79	0.82	0.70	0.14	0.86	0.78	0.76
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 8 - TF-IDF vs binary vs non-binary representation. Figures are averages. Stop words were used.

Type of n-gram representation	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F-Measure
TF-IDF	0.73	**0.81**	0.56	**0.12**	**0.82**	0.72	0.62
Binary	**0.74**	**0.81**	**0.60**	0.13	0.81	0.73	**0.66**
Non-binary	**0.74**	**0.81**	**0.60**	0.13	0.78	**0.74**	**0.66**
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 9 – Classifier performance statistics. Figures are averages. Stop words were used.

Classifier	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F-Measure
Logistic Regression	0.74	0.82	0.57	0.12	0.83	0.73	0.63
Multinomial Bayes	0.74	0.81	0.59	0.11	0.79	0.74	0.65
Random Forest	0.72	**0.83**	0.51	**0.09**	0.78	0.71	0.59
Support Vector Machine	**0.75**	0.77	**0.68**	0.19	**0.84**	**0.75**	**0.70**
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

6.2 Non n-gram results

In the Tables 10 to 15, N/A is used when it was not possible to obtain a result for a metric as doing so would have involved division by zero. In the tables below, features in red did not outperform the benchmark. The classifier used for these results is Support Vector Machine.

Table 10 - Results of punctuation-related features. Figures are averages. FP Rate = False positive rate.

Classification feature	Accuracy	Precision	Recall	FP Rate	AUC	Balanced Accuracy	F-Measure
All punctuation features combined	**0.68**	0.69	**0.61**	0.25	**0.75**	**0.68**	**0.35**
Number of capitalised words	0.65	0.68	0.46	0.19	0.7	0.63	0.2
Number of commas	0.64	**0.72**	0.39	0.13	0.67	0.63	0.16
Number of exclamation marks	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of full stops	0.56	0.6	0.18	0.11	0.6	0.54	0.02
Number of question marks	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 11 - Results of sentiment features. Figures are averages. FP rate = false positive rate.

Classification feature	Accuracy	Precision	Recall	FP rate	AUC	Balanced Accuracy	F-Measure
All sentiment features combined	**0.75**	0.71	**0.78**	0.28	**0.84**	**0.75**	**0.61**
VADER sentiment score	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Implicature	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Informativeness	**0.75**	**0.73**	0.73	0.24	**0.84**	**0.75**	0.57
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 12 - Results of grammar-related features. Figures are averages.

Classification feature	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F-Measure
All grammar features combined	**0.75**	**0.76**	0.67	0.19	**0.82**	**0.74**	**0.71**
F-Score	0.64	0.6	**0.69**	0.41	0.67	0.64	0.64
Number of adjectives	0.65	0.66	0.55	0.26	0.69	0.65	0.6
Number of adverbs	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of conjunctions	0.57	0.55	0.53	0.38	0.57	0.57	0.54
Number of determiners	0.65	0.68	0.48	0.2	0.7	0.64	0.56
Number of existential theres	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of interjections	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of modal verbs	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of nouns	0.72	0.75	0.61	0.18	0.8	0.72	0.67
Number of prepositions	0.67	0.72	0.5	0.17	0.73	0.66	0.59
Number of pronouns	0.53	N/A	N/A	**0**	0.53	0.5	N/A
Number of proper nouns	0.68	0.67	0.6	0.26	0.71	0.67	0.63
Number of verbs	0.6	0.63	0.35	0.18	0.66	0.58	0.45
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 13 - Results of lexical features. Figures are averages. FP rate = false positive rate.

Classification feature	Accuracy	Precision	Recall	FP rate	AUC	Balanced Accuracy	F-Measure
All lexical features combined	**0.77**	**0.79**	0.7	**0.17**	**0.83**	**0.77**	**0.74**
Average number of syllables per word	0.64	0.63	0.59	0.31	0.7	0.64	0.61
Average word frequency	0.57	0.6	0.28	**0.17**	0.65	0.56	0.38
Average word length	0.67	0.65	0.66	0.31	0.72	0.67	0.65
Length in words	0.69	0.69	0.6	0.24	0.76	0.68	0.64
Number of stop words	0.64	0.66	0.5	0.24	0.7	0.63	0.57
Number of words in 35 most common words in corpus	0.64	0.66	0.51	0.24	0.72	0.64	0.58
Number of words with < 5 characters	0.62	0.63	0.46	0.24	0.69	0.61	0.53
Number of words with > 7 characters	0.75	0.75	**0.71**	0.21	0.81	0.75	0.73
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 14 - Results of the feature categories for tests not involving n-grams. Figures are averages.

Feature category	Accuracy	Precision	Recall	False positive rate	AUC	Balanced Accuracy	F-Measure
All features for all categories combined	**0.8**	**0.79**	**0.78**	0.19	**0.88**	**0.79**	**0.78**
Punctuation	0.68	0.69	0.61	0.25	0.75	0.68	0.65
Sentiment	0.75	0.71	**0.78**	0.28	0.84	0.75	0.74
Grammar	0.75	0.76	0.67	0.19	0.82	0.74	0.71
Lexical	0.77	**0.79**	0.7	**0.17**	0.83	0.77	0.74
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

Table 15 – Classifier comparison for tests not employing n-grams. Figures are averages. Excludes VADAR sentiment score and F-Score, as they contain negative values, so could not be processed.

Classifier	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F Measure
Support Vector Machine	0.79	0.78	**0.78**	0.19	**0.88**	0.79	**0.78**
Logistic Regression	0.79	0.78	**0.78**	0.19	**0.88**	0.79	**0.78**
Multinomial Bayes	0.75	0.76	0.69	0.19	0.83	0.75	0.72
Random Forest	**0.8**	**0.79**	**0.78**	**0.18**	0.87	**0.8**	**0.78**
Benchmark (random data)	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

6.3 Non n-gram features combined with n-grams

Table 16 – All non n-gram features from all tests combined a unigram, versus a standalone unigram test. Figures are averages. Uses TF-IDF representation with Support Vector Machine.

Combination	Accuracy	Precision	Recall	False Positive Rate	AUC	Balanced Accuracy	F-Measure
All features plus n-gram	**0.82**	**0.82**	**0.79**	**0.15**	**0.9**	**0.82**	**0.8**
N-gram only	0.81	0.81	0.78	0.16	0.86	0.81	0.79
Random prediction benchmark	*0.51*	*0.48*	*0.49*	*0.47*	*0.51*	*0.51*	*0.48*

6.4 Results Analysis – Answering the project's questions

The sections below use the results given in the previous section to address the questions this project posed in Section 5.1 and does so in the order in which they were posed.

6.4.1 How effectively can n-grams predict formality classifications?

All the n-gram tests outperformed the benchmark to a statistically significant degree according to McNemar tests, except for some of the trigrams.

As Table 7 in Section 6.1 illustrates, unigrams are more effective overall than bigrams, which are more effective than trigrams overall. It may be that bigrams and trigrams are relatively weak features when it comes to formality classification prediction. Alternatively, it may be that, although there were formal and informal two- and three-word phrases in the corpus, they did not occur particularly often, so no strong association was made. It might therefore

be that a much larger corpus of sentences would yield a better performance by bigrams and trigrams, as the machine learning program would be more likely to know whether a 2 or 3 word phrase was formal or if it was informal.

The results also show that using unigrams in combination with bigrams and with trigrams combined produces a classification performance which is no better or worse than unigram classification performance. It may be that the extra data provided by adding bigrams and trigrams to may contain noise – that is, random data that is mistaken for a signal – and that this effect neutralises the effect of the additional useful data.

6.4.2 How do binary, non-binary and TF-IDF n-grams compare at formality classification prediction?

The results presented in Table 8 in Section 6.1 show that binary and non-binary representations produced a similar performance, as did non-binary and TF-IDF. The picture regarding TF-IDF versus non-binary is less clear. Although non-binary performed better to statistically significant extent in a McNemar test, and although it has better accuracy and recall figures, TF-IDF has a lower false positive figure and a higher AUC figure. It therefore seems that the results are too close to one another for firm conclusions to be reached.

6.4.3 What effect do stop words have on formality classification prediction performance?

Using stop words improved performance overall, as illustrated in Table 6 (in Section 6.1). All the classifiers except Support Vector Machine perform far better when stop words are included, as do the three data representations (TF-IDF, binary and non-binary). In addition, bigrams and trigrams perform far better when stop words are included. However, the performance difference is negligible with unigrams, unigrams and bigrams combined, and unigrams, bigrams and trigrams combined, as well as with Support Vector Machine.

6.4.4 How effective are the features tested other than n-grams at formality classification prediction?

Many of the non n-gram features tested proved effective, as shown in Tables 10 – 14 in Section 6.2. For example, all the lexical features outperformed the benchmark. In addition, in all instances when every feature within a category was tested in the same test, the test outperformed the benchmark (as Table 14 shows).

However, some of the features performed no better than the random benchmark. Some of the underperformance may have been due to there being small numbers of instances of certain features: there may have been insufficient data to make a well-informed classification assessment. For example, the dataset contained just 248 exclamation marks. However, that does not appear to be the case with all the features that underperformed. For example, 'number of adverbs' did no better than the benchmark, even though the 7,032 sentences had 1.04 adverbs per sentence on average. In addition, the VADER sentiment score also did no better than random predictions, even though every sentence had a VADER score. It may be that some of the classification features are ineffective, although tests on other datasets are needed before any strong conclusions can be reached, due the tests' limitations (discussed in Section 7.2).

The poor performance of implicature, shown in Table 11 in Section 6.2, is unsurprising. Although implicature is associated in theory with informal communication, as Lahiri (2015) observes, implicature is a highly subjective concept. Moreover, as Lahiri (2015) observes, with documents just a sentence in length, there is limited contextual information. As a result, the human raters of implicature have limited contextual information

6.4.5 How effective is the F-Score at formality classification prediction?

The F-Score performed significantly better at formality prediction classification than the results that would be achieved from random prediction, making it an effective feature for this purpose.

As Table 12 in Section 6.2 shows, the F-Score achieved 64% accuracy, whereas the grammar features when combined in a single test achieved 75% accuracy. This is unsurprising, as the machine learning program is likely to be more effective when it has multiple features to take into consideration (see Section 6.4.8 below for a discussion of combining features).

The F-Score achieves a better recall figure than any other grammar feature. This may be because it has a bias towards classifying sentences as formal – it has a high false positive rate (41%). However, it has an AUC score of 0.67 (when one would expect an AUC of 0.5 if the feature had no classification prediction skill), so at least some of its success in terms of accuracy is not due to this apparent bias.

Lahiri (2015) points out that Lahiri et al (2011) observed that the F-Score is unreliable on small documents such as sentences. Lahiri et al (2011) explain that this is because the F-Score is based on frequency of occurrence of various word types within a document. To illustrate, if one word in every six in a hundred-sentence document is a noun, it is more indicative of a communication that is noun-rich than if there is one noun in a standalone six-word sentence. It may therefore be that F-Score's performance as a formality classification prediction feature would have been different had documents of much longer length been used.

6.4.6 How do the four categories of non n-gram features compare in terms of formality classification prediction effectiveness?

As Table 14 in Section 6.2 shows, the lexical, sentiment and grammar categories performed equally well as each other. The punctuation category was the worst performer. Perhaps this was partly because the presence of punctuation does not show whether the punctuation is being used correctly. Suggestions for providing more effective punctuation features are provided in Section 7.3.

6.4.7 How do the four classifiers tested compare in terms of formality classification prediction effectiveness?

As Table 9 in Section 6.1 shows, for n-grams, the best performer was Support Vector Machine. For the non n-gram tests, as Table 15 in Section 6.2 shows, Logistic Regression, Support Vector Machine and Random Forest performed roughly equally well, with Multinomial Bayes being the weakest performer.

There are two fundamental causes of errors in machine learning programs: bias and variance (Hackeling, 2014). Excessively high bias is represented by inflexibility in terms of adapting to the training data, whereas excessively high variance is manifested in excessive flexibility, as predictions can be based on random noise within the data. It may be that some classifiers produce bias-variance balances that result in better overall results for formality classification.

6.4.8 Does combining formality classification features result in better formality classification prediction performance?

Combining non n-gram features into a single test usually produced a much better classification performance than using features on their own. This is illustrated in Tables 10 to 13 in Section 6.2, where most of the figures in bold, indicating best or joint best performance for the metric, are on the top line. Moreover, combining all the features in all the categories ('All Features Combined' in Table 13) produces a better set of results than can be achieved by any feature used on its own. This is possibly because the more feature data items are made available to the machine learning program for each document, the better informed its predictions are.

A McNemar test showed that combining all the non n-gram features with a unigram test did not improve formality classification prediction performance to a statistically significant degree, compared to running tests with a unigram on its own. However, as Table 16 in Section 6.3 shows, the combination of unigrams and non n-gram features did produce a marginally better performance across all the metrics than unigrams on their own achieved (though the differences are so small that they may well be due to randomness, and it may that there is a limit to the degree of formality classification prediction effectiveness that is achievable).

Chapter 7 – Conclusions

This chapter begins by discussing the project's achievements. It then explores limitations of the tests. The final three sections each suggest an area of possible future study.

7.1 What did the project achieve?

The project has achieved the aims set out in Section 1.2: it has answered the questions it posed in Section 5.1, and it has covered ground that has not been covered previously in formality classification research (which are discussed in Section 1.4).

The tests carried out have limitations (see Section 7.2 below). However, the results obtained could provide other researchers with ideas for follow-on research, and form part of a body of research which improves the understanding of formality classification prediction of written sentences. As discussed in Section 1.3, that knowledge that could potentially have practical applications.

7.2 Project limitations

Possibly the biggest challenge was the dataset not being larger. It may be that a word or phrase within the corpus is typically used in either formal or informal contexts in written English, but there are insufficient instances of it within the dataset for a strong association with either formality or informality to be made during the training phase. As a result, bigrams and trigrams may have performed better relative to unigrams if the dataset were larger. In addition, variability would have been reduced if there had been far more instances of the sentence characteristics used in the feature data. For example, there were just 33 interjections in the dataset, meaning that the test results relating to interjections are of limited use.

Another limitation is the sentences' ratings might not always reflect what most English speakers would regard as their true level of formality. The fact that there are often significant disagreements between formality annotators in studies of formality (Lahiri et al., 2011) suggests that applying formality scores to sentences is frequently not clear-cut. This situation is exacerbated by the fact that, as Lahiri et al (2011) observe, sentences offer formality raters less information to work with than larger documents do.

The corpus is sourced from blogs, from forums and from news articles (Lahiri, 2015). The way in which formality is expressed in these genres may be different to the way it is expressed in other genres, which would limit the generalizability of the results to other genres of document.

For k-fold cross validation, the value of k was 5. However, different values of k may have produced better results. For example, Brownlee (2020, August 3) states that using a k value of 10 has been found through experimentation to 'generally result in a model skill estimate with low bias a modest variance'.

The machine learning tests predict the formality classification, but not the degree of formality. Regression-based machine learning tests could be used to predict the formality scores.

Some words have multiple word types depending on their usage. For example, the word 'to' can be an adverb or a preposition (Macmillan Dictionary, n.d.). As a result, the Part of Speech tagger used, NLTK, may have sometimes inferred the incorrect word type.

7.3 Further work idea – Grammar and punctuation features

For the punctuation and grammar related classification features, the number of nouns, commas, etc in each sentence was used as a feature, rather than their frequency of occurrence within sentences. The disadvantage of the approach taken is that a high number of (for example) nouns in a sentence may simply mean that the sentence is long, rather than it being rich in nouns. Therefore, that a better classification prediction performance might be obtained from using punctuation and grammar features which also reflect the length of the sentences (such as 'nouns per word in the document').

In addition, features could be created to show whether punctuation had been used correctly. For example, there could be a feature which shows if words identified as proper nouns by NLTK are capitalised. In addition, a feature could be created to show if sentences have terminating punctuation (Peterson, et al., 2011).

7.4 Further work idea – Using 'tu' and 'vous'

Heylighen & Dewaele (1999) observe that attempts have been made to determine the formality level of French language communications using 'tu' and 'vous' (which are the informal and formal versions of 'you' respectively). The advantage of this approach is that the communicator, who is aware of the full context of the communication, is providing information about the formality of the communication. Although the current project relates to English language communications, this is an approach that could be applied to formality classification studies using languages where formality information is embedded in pronouns.

7.5 Further work idea – Using dictionary formality labels

In a formality classification study using supervised machine learning, Peterson, et al. (2011) looked up each word in a corpus of documents in online dictionaries, to check for labels such as 'informal', and used this as a basis for formality classification. Building on this approach, each word in every sentence could be looked up using an online dictionary that has words labelled as 'formal' and 'informal'. In the case of words with multiple formality classifications, the first label encountered could be taken as having the correct formality classification. The formality classification of each document would be determined by whether most words in the document are classed as 'formal' or 'informal'.

The resulting document classifications could then be compared against formality ratings provided by humans. If it can be shown that two sets of ratings are highly correlated, then that would suggest that this approach is a viable substitute to using ratings provided by humans for future studies. If it is a viable substitute, then it would allow much larger datasets to be used than would be possible if human raters were being relied upon, thereby reducing variability.

Bibliography

Abu Sheikha, F., & Inkpen, D. (2011). Generation of Formal and Informal Sentences. *Proceedings of the 13th European Workshop on Natural Language Generation*, 187-193.

Andrén, M., Sanne, J., & Linell, P. (2010). Striking the balance between formality and informality in safety-critical communication: Train traffic control calls. *Journal of Pragmatics*.

Brooke, J., & Hirst, G. (2013). Hybrid Models for Lexical Acquisition of Correlated Styles. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 82–90.

Brooke, J., Wang, T., & Hirst, G. (2010). Automatic Acquisition of Lexical Formality. *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 90-98.

Brownlee, J. (2019, August 8). *How to Calculate McNemar's Test to Compare Two Machine Learning Classifiers*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/mcnemars-test-for-machine-learning/

Brownlee, J. (2020, August 3). *A Gentle Introduction to k-fold Cross-Validation*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/k-fold-cross-validation/

Brownlee, J. (2020, August 14). *A Tour of Machine Learning Algorithms*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Brownlee, J. (2020, August 2). *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/

Chawla, K., Srinivasan, B., & Chhaya, N. (2019). Generating Formality-tuned Summaries Using Input-dependent Rewards. *Conference on Natural Language Learning*.

Chhaya, N., Chawla, K., Goyal, T., Chanda, P., & Singh, J. (2018). Frustrated, Polite or Formal: Quantifying Feelings and Tone in Emails. *Proceedings of the Second Workshop on Computational Modeling of People{'}s Opinions, Personality, and Emotions in Social Media*.

Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J., & Potts, C. (2013). A computational approach to politeness with application to social factors. *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Daoud, D., & El-Seoud, S. (2014). An Effective Approach for Clickbait Detection Based on Supervised Machine Learning Technique. *International Journal of Online and Biomedical Engineering (iJOE)*.

Davis, W. (2019). *Implicature*. Retrieved from The Stanford Encyclopedia of Philosophy (Fall 2019 Edition): https://plato.stanford.edu/archives/fall2019/entries/implicature/

Dempsey, K., McCarthy, P., & McNamara, P. (2007). Using Phrasal Verbs as an Index to Distinguish Text Genres. *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2007*.

DuBay, W. (2006). *The Classic Readability Studies.* Impact Information: Costa Mesa, California.

Fang, A., & Cao, J. (2009). Adjective density as a text formality characteristic for automatic text classification: A study based on the British national corpus. *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*.

Graesser, A., McNamara, D., Cai, Z., Conley, M., Li, H., & Pennebaker, J. (2014). Coh-Metrix Measures Text Characteristics at Multiple Levels of Language and Discourse. *The Elementary School Journal , Vol. 115, No. 2*.

Hackeling, G. (2014). *Mastering Machine Learning with scikitlearn.* Birmingham (UK): Packt Publishing Ltd.

Heylighen, F., & Dewaele, J.-M. (1999). Formality of Language: definition, measurement and behavioral determinants. *Internal Report, Center "Leo Apostel": Free University of Brussels*.

Heylighen, F., & Dewaele, J.-M. (2002). Variation in the Contextuality of Language: An Empirical Measure. *Foundations of Science*.

Hutto, C., & Gilbert, E. (2015). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014*.

Krishnan, V., & Eisenstein, J. (2015). "You're Mr. Lebowski, I'm the Dude": Inducing Address Term Formality in Signed Social Networks. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1616–1626.

Lahiri, S. (2015). SQUINKY! A Corpus of Sentence-level Formality,Informativeness, and Implicature. *CoRR, abs/1109.0069*.

Lahiri, S., Mitra, P., & Lu, X. (2011). Informality Judgment at Sentence Level and Experiments with Formality Score. *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - - Volume*, 446-457.

Li, H., Graesser, A., Conley, M., Cai, Z., Pavlik, P., & Pennebaker, J. (2015). A New Measure of Text Formality: An Analysis of Discourse of Mao Zedong. *Discourse Processes*.

Macmillan Dictionary. (n.d.). *to: DEFINITIONS AND SYNONYMS*. Retrieved from https://www.macmillandictionary.com/dictionary/british/to

Mosquera, A., & Moreda, P. (2012). SMILE: An Informality Classification Tool for Helping to Assess Quality and Credibility in Web 2.0 Texts. *AAAI Workshop - Technical Report*.

Ng, A., & Soo, K. (2017). *Numsense! Data Science for the Layman: No Math Added.* Leanpub.

Niu, X., Martindale, M., & Carpuat, M. (2017). A study of style in machine translation: Controlling the formality of machine translation output. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2814–2819.

Pavalanathan, U., Fitzpatrick, J., Kiesling, S., & Eisenstein, J. (2017). A Multidimensional Lexicon for Interpersonal Stancetaking. *Association for Computational Linguistics*.

Pavlick, E., & Tetreault, J. (2016). An Empirical Analysis of Formality in Online Communication. *Transactions of the Association for Computational Linguistics*.

Peterson, K., Hohensee, M., & Xia, F. (2011). Email Formality in the Workplace: A Case Study on the Enron Corpus. *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 86-95.

Scikit-Learn. (n.d.). *6.2. Feature extraction*. Retrieved August 22, 2020, from https://scikit-learn.org/stable/modules/feature_extraction.html

Spiegelhalter, D. (2019). *The Art of Statistics.* Pelican Books.

VanderPlas, J. (2016). *Python Data Science Handbook: Tools and Techniques for Developers: Essential Tools for Working with Data.* O'Reilly.