MOD004553 ARTIFICIAL INTELLIGENCE 2018 - 2019
AUTOMATIC TEXT CLASSIFICATION
COURSEWORK ASSIGNMENT

This assignment requires you to develop a Party Affiliation Classifier that analyses the text of a Queen's Speech and hence predicts the political affinity of the current government at the time the speech was delivered (Labour, Conservative, or LibDem/Con Coalition). There is more than one way to do this, but for this assignment you are required to implement a Naïve Bayes approach (of which there are numerous variations, the specific variation here being a Multinomial Naïve Bayes classifier). Note that on reading any of the speeches you will see that the name of the political party (or parties, if in coalition) are never mentioned, rather the party in power is always referred to as 'the government'. Whilst there is no practical need for such a Queen's Speech 'party affiliation classifier' the techniques are very relevant to other text-based requirements and is a highly topical AI subject in the field of data analytics and recommender systems. On successful completion you will have a good knowledge of a common machine learning technique applied to a specific example that can be extrapolated to many other scenarios.

In the UK, 'The Queen's Speech' is delivered by the Queen normally once a year in the House of Lords (to which members of the House of Commons and others are invited) to mark the annual state opening of parliament (usually in November but can be at other times, such as immediately after a change of government). It is written by the current government in power and is a kind of 'mission statement' by the government for the year ahead. It will typically contain proposals (whether or not achieved) for draft legislation, foreign policy objectives, and perhaps details of upcoming visits to the UK by other heads of state and any state visits to be made overseas. The length of the speech is normally around 10 minutes. There is much pomp and pageantry around the event, after which the contents of the speech are debated by MP's in the House of Commons.

Eight Queen's Speech text files are available for use for this assignment and available on Canvas; five are a training set and three are unknowns for testing purposes. This data was obtained by copying and pasting from publically available archives, in particular>
https://www.parliament.uk/about/faqs/house-of-lords-faqs/lords-stateopening/
The only pre-processing that has occurred is the removal of non-verbal content and speech marks. Although probably not required, you may if you wish extend the dataset or change the training and/or test data (a little internet searching will find plenty of on-line archive links similar to the above).

Program requirements

Given that all students should have a background in C# console programming in fairness to all, your implementation **must** conform to the following two requirements:

1. Language must be C#
2. Application must be console based (a GUI interface would be an unnecessary distraction to the AI aspect of this work).

Program functionality
Your program should do the following:

1. On start up the program should prompt the user to either a) undertake training or b) undertake a classification. If training is chosen, the user must specify the relevant input source code file(s). After training the program should optionally write the Bayesian network to file and default to the classification option. If classification is chosen, the user must either use the current network created by the prior training phase or read a pre-trained network from file. The user then enters the filename of the document to be classified, and a result is displayed back to the user.
2. Appropriate statistics concerning training and classification should also be presented.

Algorithmic Development
A basic procedure for implementing the Multinomial Bayesian Network is described below; the theory is not presented here. A trained Bayesian Network is a set listing of unique textual terms and their corresponding conditional probabilities for a series of categories. Further information including relevant theoretical background is listed under 'References'.

Step 1: Determine the Prior Probabilities
The number of documents in the training set (Tdocs) will contain various documents attributed to the various designated categories ($Tcat_a$, $Tcat_b$, etc), i.e.

$$Tdocs = Tcat_a + Tcat_b + \ldots Tcat_n$$

The prior probabilities will therefore be;

$$P(cat_a) = Tcat_a / Tdocs$$
$$P(cat_b) = Tcat_b / Tdocs$$
…
$$P(cat_n) = Tcat_n / Tdocs$$

Step 2: Calculate the conditional probabilities
First count the total number of unique words throughout the training documents (Nwords), and the total number of words in each category, including repeats ($Ncat_a$, $Ncat_b$, etc). Create a table for each category. For each table list in the first column all the unique words present in the training set, and in the second column their respective frequencies summated from the documents for that particular document category, fcat[word]. The third column for each table should then list the conditional probabilities $P(word_i/cat_n)$ of the words in that category; for example for table category 'a' for a particular word this would be

$$P(word/cat_a) = (fcat_a[word]) + 1) \;/\; (Ncat_a + Nwords)$$

If there are two categories, two tables will be constructed with identical dimensions. If there are six categories, then six tables of data need to be constructed etc. Typically there will be a small number of columns and a larger number of rows, the latter number dependent on the spectrum of unique words in the documents.

You will probably be aware at this point that this assignment fundamentally requires the parsing and extraction of words from text and the manipulation of array data structures.

Step 3: Classify an unknown document
Now we can analyse a document (doc) whose category is unknown and attempt to classify it. The principle here is that the training set tells us the pattern of features for each class. By reverse engineering what is essentially the Bayes formula and summating the conditional probabilities for each word for each category the highest score will indicate the classification. In other words for each category, one calculates the product of the derived training set conditional probabilities for each word occurrence in the document (ignore any new unique words), finally multiplying it by the prior probability, i.e.

$P(cat_a/doc) = P(word1/cat_a) \times P(word2/cat_a) \times \ldots P(word_i/cat_a) \times P(cat_a)$

$P(cat_b/doc) = P(word1/cat_b) \times P(word2/cat_b) \times \ldots P(word_i/cat_b) \times P(cat_b)$

Etc..

For example, if the document contained 189 words, then there would first need to be 189 probabilities as read from the column of conditional probabilities in a given category table multiplied together (or less if any new words are present; such new words are ignored). That value is then multiplied by the prior probability of that category to get the probability of that document belonging to the given category. If there were four categories, a comparable calculation would need to be undertaken 4 times etc and the category with the highest value will indicate the class the document belongs to. Note because these multiplications often generate small final values it is possible to generate floating-point overflow errors, so it is better to take the logs of the probabilities and sum them, so that the highest summed log values of the categories will classify the document.

More generally, implement the assignment in stages e.g. get a program/method to read and store the words; implement a method to count and store the frequency of a word etc. Backup working solution states regularly and do not try and extend the program without debugging each stage of improvement as otherwise the program could become impossible to debug.

Enhancements and simplifications
You may feel you need to limit the functionality of the program to prioritise a working solution, or you may have time to further develop the program.

Simplifications are mainly achievable around file input and output. For example you could a) hard code in the content of the training text files, training the Bayesian network every time the program is run thus avoiding reading the training documents from file b) do not implement the writing to file of the Bayesian network c) do not implement the reading from file of a pre-trained Bayesian Network.

There are many enhancements that can be done to improve the basic model. In order of difficulty some of these techniques are:
Removing stopwords. These are common words that do not really add anything to the classification, such as ‘a’, ‘able’’, either’, ‘else’, ‘ever’ etc. So for example *The election was over* could reduce to *election over*. A file containing common stop words is included on the Canvas delivery site for this purpose if required.
Lemmatizing words. This is the grouping together of different inflections of the same word. So *election*, *elections*, *elected*, etc could be grouped together and counted as more appearances of the same word.

Using n-grams. Instead of counting single words as described here, we could count sequences of words, like "clean match" and "close election". This is the concept of 'word families'.
Using Term Frequency Inverse Document Frequency. Instead of just counting frequency a more advanced technique could be used such as TF-IDF to penalize those words that appear most frequently in most of the texts.

Assessment

The automatic text summarization assignment is component 010 of the assessment and contributes 100% of the overall module mark.

Submit your work to the iCentre or equivalent as a short report by Friday 14th December 2018 (end of teaching **week 12)**. Ensure the removable storage medium is adequately secured within or to your report, which should contain;

1. Source and compiled code on a CD/DVD or USB stick. This media should also include a copy of *all* your submitted work for this assessment to comply with the University requirement for submission of all assessed coursework in electronic form.
2. Documentation to include a 600 word evaluation of the classification tool (for example, design of any classes and/or data structure(s), strengths and limitations of the implementation). You are *not* expected to undertake or present the results of formal testing (test plan, test table, etc).
3. Include instructions for running the program in an appendix.
4. Include a copy of your source code in an appendix.
5. A hard-copy print out of the report including elements 2-4.

Your final mark will be an amalgamation of four equally weighted components. See the separate mark scheme/ mark sheet for details on how this assignment will be marked. *Note that even if your final program is not fully functional (or even will not compile) you can still get credit for the way the program was designed, the coding and the evaluation.* Feedback will be available in 3 ways a) via a personalised completed mark sheet attached to each assignment on collection of the coursework b) via an email sent to each student which will contain personalised content summarising the results c) via e-vision accessible by students listing percentage, grade and outcome (subject to assessment board rules).

Reassessment

If you are unfortunate enough to require a coursework reassessment, you should attempt this coursework assignment again. Be aware that there is more than one approach to achieving a solution and you are encouraged to explore a different approach compared to the first attempt.

References

Manning, C.D., Raghavan, P. and Schutze, H., Introduction to Information Retrieval, Cambridge University Press, 2009. Ch. 13. Text classification and Naive Bayes. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf
McCallum, A. and Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. In: Proceedings of AAAI, 1998. http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf
Sebastiani, F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 34(1):1-47, 2002. Digital library: https://dl.acm.org/citation.cfm?id=505283

MOD004553 Coursework feedback sheet 2018-19 Sem1							
Student	**Functionality**	**Program design**	**Enhancements**	**Evaluation, code clarity and presentation**	**Comment**	**Overall 010 mark %**	**Overall module grade**
	Does program execute, proceed to completion, generate a run-time error or crash? Interface clarity. Successful training / classification? File processing. Output result(s).	Choice of user-written classes and their methods including; file processing, strategies taken for text processing, word list / data / table / array data structure construction.	To what extent does the code implement the main requirements? Choice and complexity of enhancement(s) and relevance.	Assessment of achievements and limitations. Report layout and structure. Clarity of code layout and in-line comments. Physical presentation. Any items missing?	Personalized comments specific to student submission.	Each component is assigned a grade (F, D, C, B, A) and each maps to a mark (0-100) from which the average is taken.	F (<40); D (40-49); C (50-59); B (60 - 69); A (70+)
Sid number	**Grade**	**Grade**	**Grade**	**Grade**	**Comment**	**Final Mark**	**Final Grade**