DATA8001 - Assignment 2

Student ID: R00207204

Student Name: Bernard McNamee

Report

Summary

Assignment : *Create 3 multi-class classification models to classify news article categories using the sample data provided to train & test your models.*

Process

1. ETL – 2000 text file documents were imported, filtered by html tag, and content features (headline, article and category) saved to related columns in a Pandas dataframe.
2. Explore – a statistical summary and plots shows the data is quite uniform (word count and document category count) with no missing or incomplete features.
3. Pre-process – in preparation for running ML models, the text was 'cleaned' again removing unwanted punctuation, digits, uppercase and stopwords
4. Tokenise – inflected words were reduced to smaller units called token by one of two functions, Stem and Lemmatise, eg caring → car or caring → care
5. Vectorise – the remaining words were converted to float numbers using a frequency–inverse document frequency TF-IDF method. TF-IDF was preferred over Bag Of Words BOW as BOW vectorises the count value of words in the document, whereas TF-IDF vectorises a selective statistical value based on it's importance to the document within the corpus.
6. Model – 3 multiclass classifcation models were selected
    1. K Nearest Neighbour (KNN)
    2. Support Vector Machine (SVM)
    3. Multinomial Naive Bayes (MNB).
7. Tune Model - Grid Search was applied to each model to discover the best hyperparameters and values.

8. Model – the data was split into training and test, and each of three models, trained and tested. Model accuracy was calculated including classification reports and confusion matrices.

9. Model selection – each model was run with various parameters fixed one at a time and the results written to file for compilation in order find the best model.

Conclusion

The ML algorithms, KNN, SVM and MNB, were selected based on suitability for multiclass classification text problems with small datasets.

KNN hyperparameter k (or n_neighbors) was evaluated for a range of values and plotted against accuracy score to find the best value k =38. Similarly, SVM hyperparameter C was reduced from 0.0001 to 0.001 reducing the training model accuracy from 99% to 91% to avoid over fitting – this increased the margin between classes so the model would work better on unseen data.

Although it requires a knowledge of the initial hyperparameters, Grid Search worked well in identifying the best hyperparameters to models:

- KNN (metric='euclidean', n_neighbors=38, weights='distance')
- SVM (C=0.0001, degree=2, gamma=100, kernel='poly', probability=True)
- MNB (has few/no hyperparameters)

Besides, hyperparameters, Stem tokenisation method was found to be better then Lemmitisation, and features 'Article' was found to be better than 'Article' + 'Headline' or 'Headline'.

The best algorithm was found to be SVM with an average weighted F1 score of 92%. The KNN and MNB models were close with 87% and 89% F1 scores.

For my next ML project, I would like to a better understand algorithm selection - learn how each works and suitability for particular tasks – and, also, how to select hyperparameters with a view to tuning models.

Question 1 - Why do you think that AI projects have such a poor rate of success?

(max 300 words)

Bayesian logic prompts the conditional probability question, do software projects, never mind business projects in general, have a poor rate of success? The answer is yes, they do. However, a more insightful analysis will focus on reasons why, specifically just AI software projects, so often fail and skip past the general answer and reasons such as unclear objectives, lack of leadership support or poor communications.

Lack of expertise – AI is a new and emerging field. Companies may have inhouse traditional software development expertise but not are not familiar with AI tools and hardware. Data science is a buzzword, analysts in short supply command high salaries, and *"anyone who has worked in data analytics or software development who has done some sample data science projects are labeling themselves as data scientists after taking a short course online." (Ref Dzone)*

Data quality – AI projects require a large volume of data, the more data available, the better the predictions. Frequently, data is siloed in various departments requiring a significant effort to merge the data from multiple sources. After building a merge pipeline, data needs to be labeled, another significant undertaking – this requires tools or manpower. Both tasks, merging and labeling, need to be done right for the historical and operational data – bad data will produce results that aren't actionable or insightful. It's difficult not to underestimate the costs of these functions which risks adding to project cost, time and goodwill.

Unrealistic Expectations - the cost of AI projects tends to be extremely high leading may companies to target overly ambitious moon-shot projects. These project goals will push deadlines out and push the data science team to their limits. The business leaders risk losing confidence and canning the project. Alternatively, a focus on just one important business problem is a more realistic strategy while building expertise, systems, and overseeing a culture change with regards to data and user management.

Question 2 - Discuss the difference between Schema on Read vs. Schema on write architectures with examples.

(max 300 words)

Schema on read is the new database architecture replacing traditional RDBMS relational databases designed with schema on write architecture when the business application running on RDBMS systems have reached their design limitations. RDBMS systems often served by SQL applications were designed for structured data, eg accounting, whereas NoSQL systems (often used to denote Schema on read systems), eg Twitter, must serve unstructured data at much greater scale - low latency, high volumes and high transaction volumes.

To address the scalability and performance wall, companies with these application requirements initially threw money at the RDBMS problem. They invested more hardware and upgraded faster hardware. They hired more database engineers and focused on technical tweaks particularly regarding the database design. However, the application demands outstripped this progress so they started to develop inhouse solutions such as Facebook's Casandra and Amazon's Dynamo.

Since then off the shelf database applications such as MongoDB have reached the marketplace. MongoDB and others provide address the RDBMS limitations and offer flexible data models, horizontal scaling, fast queries and are easier for developers.

Flexible data models allow a flexible schema change as design requirements change. Horizontal scaling enables commodity hardware to be used – this is cheaper than vertical scaling which requires more expensive hardware upgrades demanded by RDBMS. Fast queries - RDBMS data is typically normalised and 'joined' to multiple tables and as tables grow the query speed slows wheras NoSQL does not requires 'joins' and data is sited where it is accessed so data storage is optimised for queries. Easier for developers - NoSQL databases map their data structures to those of popular programming languages allowing developers to store their data in the same way that they use it in their application code leading to faster development time and fewer bugs.