Text Mining applied to Algorithm Selection

JT Anibal, LN Phan, DV Le, IJ Post, IS Withrow, AK Peltekian

Introduction: Why we did it

Our goal is to make machine learning available to researchers & students, regardless of field

Introduction: MLAI

- We want our clients to use MLAI without any knowledge of the data or any knowledge of programming
- One platform for all data uploads
- Integration of many different tools/algorithms through informal partnerships with researchers
- Consistent selection of reasonable approaches from a wide variety of options
- Clear explanation of how results were obtained & accessible resources for further education and theses/publications

Introduction: MLAI

To provide reasonable analysis for so many different data possibilities, we need text mining...

User Input: The First Step

- User must input a text description of the data
- MLAI will parse this input and identify key words and bigrams (two word phrases)
- We determined that nouns, noun-adjective bigrams, and noun-noun bigrams give the best, most reliable performance
- To isolate these words, we use the spaCY pos-tagger

User Input: Synonyms

- We use spaCy package to generate synonyms of more common words from built-in corpora (i.e. image, picture)

- These synonyms are added to our original set of key words and bigrams

- This makes our text mining results more generalizable and consistent

Not following directions...

- When the user does not provide any description, this lessens the probability of a viable approach

- If this occurs, MLAI defaults to **Gradient Boosted Classifier** and **Support Vector Machine** (sklearn) if the analysis is supervised

- If the analysis is unsupervised, MLAI defaults to **K-Means clustering** and **Hierarchical Clustering** (also sklearn)

- In general, the better the description, the better the performance

Text Mining: Finding the Texts to Mine

- We mine the science direct database through the Elsevier Developer API and the Elsapy python package
- The search queries are made using the user input (without synonyms) and the universal term “machine learning”
- Bigrams are combined with the universal term (example query: “handwritten digits machine learning)
- For single words, all possible two-word combinations are derived (i.e. ‘immunocompromised cytometry machine learning”)
- ScienceDirect returns the papers that match most strongly with the set of keywords in the query

Text Mining: Finding the Texts

With even a moderately detailed description, we obtain papers relating to many crucial aspects of the study!

Text Evaluation: Format

- The content of the papers are returned as JSON files and converted to Python dictionaries
- The number of papers mined depends on the time-related needs of the user
- Each section of the paper (i.e. abstract) is saved as a single Python string
- Strings that do not meet a certain length threshold (i.e. author list, acknowledgments) are deleted
- Using string pattern matching allows MLAI to perform lemmatization, further improving the generalizability of results

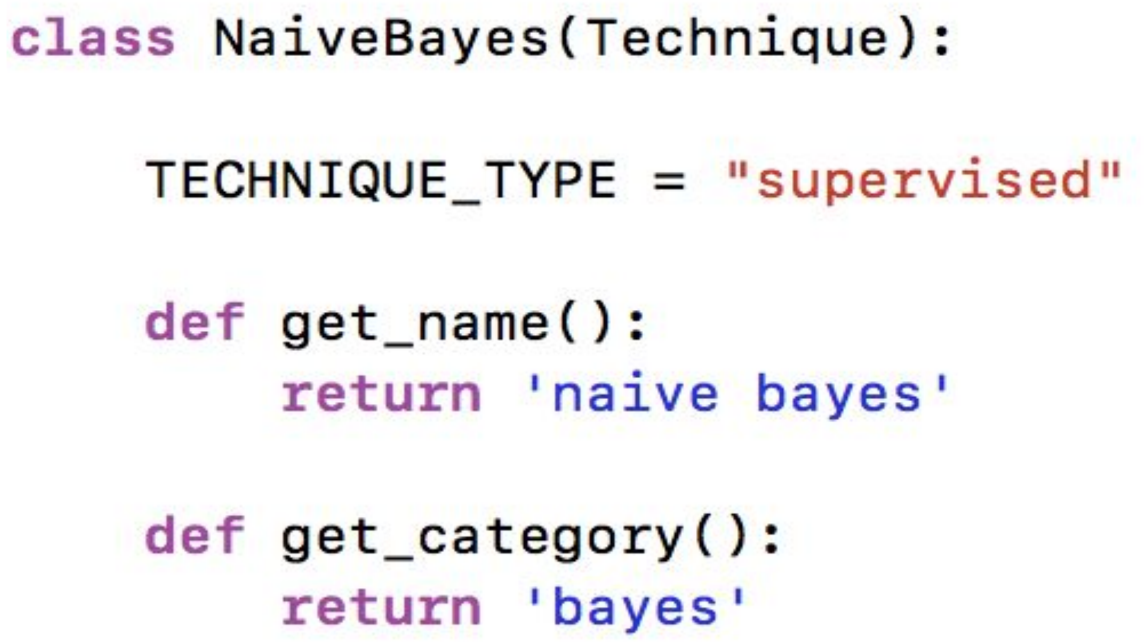
Example: This is one string

“We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective.”

Text Mining: What to search for

- Obviously, we want to find the user keywords in the papers
- We also need to look for ML techniques
- Each technique in our MLTechniques library has two string representations: “category name” and “specific name”
- We also search for these technique names in papers

Example: We search for bayes and naive bayes



```
class NaiveBayes(Technique):

    TECHNIQUE_TYPE = "supervised"

    def get_name():
        return 'naive bayes'

    def get_category():
        return 'bayes'
```

Text Evaluation: Neighbourhood Scoring

- Breaking the paper down by sections also allows for **neighbourhood scoring**

- Important words are only counted **once per section**

- This allows MLAI to evaluate the true prominence of technique throughout a study

- MLAI will not become biased by words that are excessively repeated in discussion or further work sections

Example of what we want to minimize...

This section is not actually about the study, but uses keywords repetitively

“When surveying other image classification methods, we found that our method out performed a variety of common techniques for detecting images of handwritten digits. For example, a linear SVM classified the MNIST dataset (handwritten digits from 0-9) with an AUC of 0.94, whereas our technique, ______, yielded a performance of 0.98 with comparable time complexity. In the past, SVMs and convolutional neural networks (CNNs) have been the premier methods for image classification, but we believe our method has successfully combined the ideal features of both methods.”

- 2 occurrences of SVM irrelevant to the actual paper
- 2 occurrences of CNN irrelevant to the actual paper
- 3 occurrences of images irrelevant to the actual paper
- 2 occurrences of handwritten digits irrelevant to the actual paper

Text Evaluation: Scoring

- For each section of the paper, we first count how many of the distinct technique words appear

- For each section of the paper, we first count how many of the distinct “technique” words appear (i.e. naive bayes)

- We then check how many distinct user inputted words/bigrams are in that section

Example: Potential Keywords highlighted

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called “dropout” that proved to be very effective.

Text Evaluation: Scoring

For each paper, the score for each technique in the paper is given a score. This score is added to the total score and averaged after all papers have been mined

#1 - Score first assigned based on how many sections of the paper contain the name of that technique (representing universality)

#2 - Score is adjusted by the percentage of found user keywords vs. the total number of user keywords from the original description

#3 - Score is adjusted for any found bigrams (bigrams are specific, indicating a better match)

#4 - Scores for specific names (naive bayes) are given extra weight compared to general names (bayes)

Example #1

This…

“We found that our CNN architecture performs well on images of handwritten digits compared to other studies”

Scores higher than this…

“We found that our CNN architecture performs well on large image datasets compared to other studies”

This is because we found more keywords AND a bigram - both increase score

Example #2

This…

“We found that our novel implementation of a naive bayesian framework performs well on X data compared to other studies”

Scores higher than this…

“We found that our novel implementation of a bayesian framework performs well on X data compared to other studies”

This is because we a specific name - we want to be as targeted as possible

Final Selection

- Select the top quartile of scores after all papers have been mined and all scores have been adjusted
- We want two approaches to train on the data
- If a specific technique name is in the top quartile, select that immediately
- If we have to choose from the categories (i.e. different CNN architectures), further selection procedures are needed

Final Selection: Choosing from categories

If we have to choose between categories, we use metrics such as:

#1 - expected algorithm performance based on past uses

#2 - how many times has the algorithm been used

- If the total number of uses across all techniques in the category is **below a set threshold**, technique(s) will be selected stochastically

- Even narrowing down to category still gives reasonable approach (i.e. if random forest is optimal, decision tree is likely still reasonable)