Predictive Coding

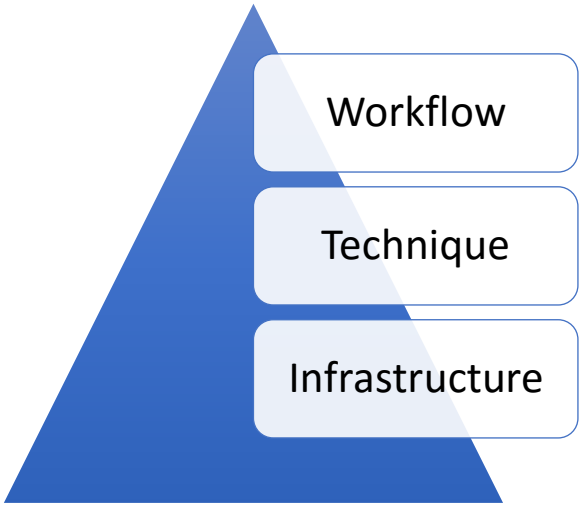
Background

Predictive coding (or Technology-assist review) uses machine learning and natural language processing (NLP) techniques against the gigantic datasets of forensics and eDiscovery. Predictive coding is typically used to replace or supplement document review processes for identifying responsive, privileged, confidential or other document categories. Although there has been substantial growth in the use of predictive coding over the last few years, like any other technologies, it's not guaranteed to perform well without the proper configuration and settings. It needs to be assisted and trained with a deep understanding of the data, domain, and the technology itself, making it quite difficult in some cases and not normally a straight out-of-the-box technology.

In the next section, you will see how Vista Analytics opens the black box of predictive coding to our customers and developed a solid predictive coding solution leveraging both machine learning and cloud computing. Our solution is not only more accurate than many competitors but also more cost-effective than other solutions.

Predictive Coding in Vista Analytics

From our perspective, a successful predictive coding solution consists of three fundamental elements: workflow, technique, and infrastructure. Workflow defines how to prepare training, validation and scoring datasets for model training, evaluation and scoring. Technique is about the appropriate Natural Language Processing (NLP) and machine learning techniques for a given scenario. Infrastructure includes where to implement the solution and what resources a solution can leverage.



Workflow

Without a solid workflow, predictive coding can very easily fail or provide sub-optimum results.. One of the dangers in inherent in predictive coding is the unseen nature of many types of errors. These errors can lead to substantial effort by attorneys in reviewing incorrect documents. In Vista Analytics, we avoid the pitfalls by following a solid workflow.

- **Document Preparation:** Documents in a predictive coding exercise may come in with various formats or encodings. It's very important to convert them to a text (.txt) format and make sure that they are all usingUTF-8 encoding.
- **Training and Validation Dataset Setup:** All collected documents will be divided into 3 parts: training set, validation set, and scoring set. With attorney's help, labels are assigned to documents in training and validation set. That means we know exactly if a document is relevant or irrelevant in the training and validation sets. Depending on the Use case and data availability, training and validation sets are usually small with a few thousand documents minimizing the cost and effort involved in the manual review.The size of the training and validation sets and the ratio of relevant and irrelevant documents inside are determined by statistical analysis.
- **Model Training and Tuning:** Models are trained using 10-fold cross-validation in the training data set. In a 10-fold cross validation, the training set is randomly partitioned into 10 equal sized subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the model, and the remaining 9 subsets are used to train a model. The process is repeated 10 times, with each of the 10 subsets used exactly once as the validation data. We take an extra step to validate the trained model on the hold-out validation set to make sure it will perform well on the scoring set. The best model is chosen based on its Area Under the Curve (AUC) in 10-fold cross validation. The goal of the cross validation is to assess the predictive accuracy of the training sets to the larger population of documents.
- **Scoring and Reporting:** After scoring a large population of documents with previously unknown labels, we output the results to a delimited file (normally a .csv format) and generate descriptive statistics based on customers' needs.

Please contact Vista Analytics for more details about our considerations of workflow setup and the statistics behind it.

Technique

Like any machine learning problem, the keys of accurate predictive coding are: feature and algorithm. In Vista Analytics, we open the black box for our customers and make these two steps as transparent and configurable as possible.

The first step of predictive coding is to represent each document by a term vector. Each element in a term vector represents a word or phrase. Some important decisions need to be made to generate term vectors, which have huge impact on the performance of the predictive coding model. Below we list some but not all settings that our solution will allow you to configure:

- **Binary vs. Term Frequency (TF) vs. Term Frequency-Inverse Document Frequency (TF-IDF)**: As introduced above, each element of a term vector represents a word or phrase. The value of an element can be 0/1 (binary) or a numeric value between 0 and 1 (TF or TF-IDF). In the binary setting, zero means a word is absent in a document while one means the opposite. In the TF or TF-IDF setting, the value of an element tells not only the existence of a word given a document, but also its importance to a document. Specifically, in the TF setting, a vector element represents the term frequency of a word in a document. The higher the frequency, the larger the value, and the higher importance of the word to a document. TF-IDF is a variant of TF, which plays down the importance of a word if it occurs in many documents.
- **Feature Number**: In predictive coding, each word or phrase represents a feature. When we discuss the number of features to use, we refer to the length of these term vectors. In most cases, it is not a good idea to consider all words or phrases that exist in a corpus when building a predictive coding model. Since word frequency follows the power law (a relative change in one quantity results in a proportional relative change in the other quantity) in natural language, many words occur too infrequently to be useful in building predictive coding models. Thus, it is important to have some controls on the number of features to use.
- **N-gram**: In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. In predictive coding, we consider each word as an item, so that 1-gram (unigram) means using each individual word as a feature while 2-gram (bigram) means using two consecutive words as a feature.
- **Stemming**: In natural language processing, stemming is the process of reducing inflected (or sometimes derived) words to their word stem or root form. For example, a stemming algorithm will reduce the words “fishing”, “fished”, and “fisher” to the root word “fish”. The advantage of stemming is to largely reduce the number of features with the assumption that it is sometime meaningless to treat a word and its plural form as separate features. However, stemming is not always necessary and it might introduce noise in some cases.

Once the feature vectors are generated, the key is to select the right machine learning model and parameter to build the classifier. Logistic regression and Support Vector Machine are two of the best off-the-shelf machine learning models that are widely used in predictive coding. In our solution, users can train different classifiers with different parameters and arrive at the best model and setting using a grid search strategy. Assuming that there is a single best classifier for all scenarios is, we believe, one of the failings of current predictive coding solutions in the legal marketplace.

Infrastructure

The Vista Analyticssolution is implemented based on Apache Spark and Amazon Elastic MapReduce (EMR). Apache Spark is a fast and flexible engine for big data processing, with built-in modules for machine learning. Apache Spark consists of a collection of state-of-the-art machine learning models which allow us to build predictive coding models using the best technique for a given scenario. EMR is an Amazon Web Service (AWS) for large volume data processing and analysis. It uses Hadoop, an open-source framework, to quickly and cost-effectively process vast amounts of data. By combining the power of Apache Spark and Amazon EMR, we can train dozens of predictive coding models in parallel, evaluate their performance and pick the best model. In addition, applying the module to millions of documents can be accomplished in minutes as opposed to days or weeks.

Use Case

In this scenario, we demonstrate the use of predictive coding to separate spam emails from non-spam emails, which, to some extent, is similar to differentiate relevant and irrelevant documents in eDiscovery. The use of spam/non-spam emails in this analysis was based on the availability of this data set as opposed to confidential data sets used in Discovery.

Specifically, we used CSDMC2010 SPAM corpus in our experiment. This dataset contains 4327 labeled emails out of which there are 2949 non-spam emails and 1378 spam emails. We randomly split this dataset into training and validation sets. We then built and fine-tuned our model on the training dataset using 10-fold cross-validation, to further investigate the effectiveness of our model in the hold-out validation set.

As discussed in the previous section, we have opened the black box of predictive coding and make it as transparent as possible. We launched an Amazon EMR cluster and leveraged the Spark Pipeline API to build multiple predictive coding models with different settings in parallel. Benefits of using Spark include but are not limited to: 1) models with subtle configuration differences may result in significantly different performance; 2) Spark is flexible and quick to build and compare different models and allows us to customize our solution fully driven by the specific data characteristics; 3) parallel computing in EMR enables us to quickly and cost-effectively train models which can take days in a non-parallel computing environment.

Table below shows the 10-fold cross validation AUC of our predictive coding models under different settings in the training dataset. Here, we tested binary term vector vs. term-frequency term vector, using unigram (N=1) and bigram (N=2), using term vector with 1,000 words, 5,000 words or 10,000 words, and using different regularization parameters in the logistic regression model. As you can see, the best setting of parameters produced a model with nearly 0.998 AUC (highlighted in green), while the worst setting of parameters yield a model with only 0.937 AUC (highlighted in red). With this relatively simple spam email classification problem (spam email detection has been widely used), the best setting can outperform the worst setting by nearly 6.5%. This demonstrates the importance of trying different settings to pick the best model, because the boundary between relevant and irrelevant documents ismuch fuzzier in forensic investigations and eDiscovery.

Table 1. 10-fold cross validation AUC of different models in spam email training data

Binary vs. Term-Frequency	Regularization Parameter λ	N-gram	# of Features	Area Under the Curve (AUC)
Binary	0.1	1	1000	0.9890235003403679
Binary	0.1	1	5000	0.9957067798374365
Binary	0.1	1	10000	0.997271877226154
Binary	0.1	2	1000	0.9538280090469424
Binary	0.1	2	5000	0.9818555815554528
Binary	0.1	2	10000	0.9890945217810384
TF	0.1	1	1000	0.9823678980708958
TF	0.1	1	5000	0.9920637627969465
TF	0.1	1	10000	0.9932668135663414
TF	0.1	2	1000	0.9400254137546925
TF	0.1	2	5000	0.9746656278968298
TF	0.1	2	10000	0.9839525437080279
Binary	0.01	1	1000	0.9875699960243376
Binary	0.01	1	5000	0.9955929255861299
Binary	**0.01**	**1**	**10000**	**0.9975319606719455**
Binary	0.01	2	1000	0.9456864519599613
Binary	0.01	2	5000	0.9790730930254945
Binary	0.01	2	10000	0.9879151419108348
TF	0.01	1	1000	0.9816236989125003
TF	0.01	1	5000	0.9913451589015764
TF	0.01	1	10000	0.9918196837464893
TF	**0.01**	**2**	**1000**	**0.9369406158310267**
TF	0.01	2	5000	0.9705260298878629
TF	0.01	2	10000	0.9810760879088167

For further validation, we applied the best model (TF, Unigram, feature vector with length equals to 10000, and λ=10,000) on the hold-out validation set. AUC of our model in the validation set is nearly 0.9988 - almost perfect! Furthermore, different from some predictive coding solutions in the market, which only return a binary prediction (either relevant or irrelevant), our solution returns a numeric value between 0 and 1. The higher the value, the stronger the confidence that our model believes a document is relevant. This allows for the results of the predictive coding exercise to be used in multiple ways. For instance it can be used as a junk filter, or a prioritization tool, a QC tool or a host of other options. We summarize the prediction in a pivot table below. From the table, we can see that by reviewing all emails with score larger than 0.8, we can find 99% of the relevant emails without any misclassification.

Model Output	True Non-Spam Email	Predict Non-Spam Email	Accumulated Non-Spam (%) Email
$0.9 < x \leq 1$	487	487	97.4%
$0.8 < x \leq 0.9$	8	8	99%
$0.7 < x \leq 0.8$	2	3	99.4%
$0.6 < x \leq 0.7$	0	2	99.4%
$0.5 < x \leq 0.6$	1	7	99.6%
$x \leq 0.5$	2	0	100%