CASE STUDY

Understanding Predictive Coding Workflows using Intella Connect

Vound
AMERICA • ASIA • EUROPE

Intella predictive coding

Contents

Introduction

Since Judge Andrew Peck's 2012 ruling in ***Da Silva Moore v. Publicis Groupe & MSL Group*** approving the use of computer-assisted review, predictive coding has been considered a court approved protocol for managing review in eDiscovery. In many cases with many document collections, predictive coding is considered a more efficient method of conducting review because it doesn't require "eyes-on" review of all of the potentially responsive documents to complete responsiveness determinations. It's also considered as accurate, if not even more accurate, than manual review in many cases – not only saving time and cost during document review but doing so without sacrificing quality.

However, many legal professionals have not embraced predictive coding for their cases when it could be used to conduct the review more efficiently and cost-effectively. Why? In many cases, the workflows and the technology associated with predictive coding aren't intuitive to them, so many of them tend to stick with the keyword search and manual review processes with which they've become comfortable. As a result of not understanding how predictive coding works, they miss the potential benefits that predictive coding technology and workflows can bring to their cases. It is human nature to avoid what we don't understand.

To address the lack of understanding of the process, this white paper walks through an example of a predictive coding workflow using a well-known public domain collection of data to demonstrate the accomplishment of a review objective regarding that data collection. This demonstration is being conducted using the predictive coding module within *Vound's Intella® Connect* browser-based review platform. The white paper consists of five main sections:

- **Understanding the Data Set Used for Predictive Coding Exercise**: Discuss the data set to be used and the goal of the predictive coding exercise to be achieved;
- **Identifying the Starting Candidate Data Set**: Discuss the approach used to identify the documents to be used within the predictive coding exercise;
- **About Intella Connect Predictive Coding**: Understanding the workflow associated with Intella Connect's predictive coding module;
- **Predictive Coding Exercise with Candidate Data Set**: Walk-through example of the predictive coding process with the actual identified candidate data set; and
- **Conclusion**: Final comments regarding the predictive coding workflows and potential time and cost savings through a predictive coding workflow with Intella Connect.

Understanding the Data Set Used for Predictive Coding Exercise

The data set used for this particular predictive coding exercise is the Enron Data Set associated with the Enron scandal of the early 2000's. This data set was chosen because it is a sizable public domain collection with a variety of content, making it a terrific representative example of how a typical litigation document collection might look.

Enron Scandal

The Enron scandal was an accounting scandal involving Enron Corporation, an American energy company based in Houston, Texas. Upon being publicized in October 2001, the company declared bankruptcy and its accounting firm, Arthur Andersen – then one of the five largest audit and accountancy partnerships in the world – was effectively dissolved. In addition to being the largest bankruptcy reorganization in U.S. history at that time, Enron was cited as the biggest audit failure.

Enron's use of mark-to-market accounting (accounting based on market value, which was then inflated), off-balance-sheet vehicles and complex financing structures created confusion among shareholders and analysts. Eventually, the specifics about Enron's accounting practices came to light, leading to public disclosure of accounting irregularities by both Enron and Arthur Andersen, leading to the aforementioned bankruptcy and dissolution of the two companies.

Enron Special Purpose Entities

One of the specific areas of accounting irregularities that were ultimately disclosed was Enron's use of special purpose entities. Enron used special purpose entities—limited partnerships or companies created to fulfill a temporary or specific purpose to fund or manage risks associated with specific assets. The company elected to disclose minimal details on its use of "special purpose entities". These shell companies were created by a sponsor but funded by independent equity investors and debt financing. For financial reporting purposes, a series of rules dictate whether a special purpose entity is a separate entity from the sponsor. In total, by 2001, Enron had used hundreds of special purpose entities to hide its debt. Examples of special purpose entities included[1]:

- **JEDI and Chewco**: In 1993, Enron established a joint venture in energy investments with CalPERS, the California state pension fund, called the Joint Energy Development Investments (JEDI). In 1997, Enron asked CalPERS to join Enron in a separate investment. CalPERS was interested in the idea, but only if it could be terminated as a partner in JEDI. However, Enron did not want to show any debt from assuming CalPERS' stake in JEDI on its balance sheet, so they developed the special purpose entity Chewco Investments, a limited partnership which raised debt guaranteed by Enron and was used to acquire CalPERS's joint venture stake for $383 million. Because of Enron's organization of Chewco, JEDI's losses were kept off of Enron's balance sheet. Chewco was named after the Star Wars character Chewbacca because it was created to hide losses from the Joint Energy Development Investment Limited, known by its acronym "JEDI". Like Chewbacca, the Jedi Knights were prominent characters in Star Wars.
- **LJM**: In 1999, Enron formulated two limited partnerships: LJM Cayman. L.P. (LJM1) and LJM2 Co-Investment L.P. (LJM2), for the purpose of buying Enron's poorly performing stocks and stakes to improve its financial statements. The LJM companies were named by CFO Andy Fastow, based on the names of his three kids (Lea, Jeffrey and Matthew). LJM 1 and 2 were created solely to serve as the outside equity investor needed for the special purpose entities that were being used by Enron.
- **Raptors**: Enron transferred to "Raptor I-IV", four LJM-related special purpose entities named after the velociraptors in Jurassic Park, more than "$1.2 billion in assets, including millions of shares of Enron common stock and long-term rights to purchase millions more shares, plus $150 million of Enron notes payable" as disclosed in the company's financial statement footnotes.
- **Whitewing**: Whitewing was the name of a special purpose entity used as a financing method by Enron. In December 1997, with funding of $579 million provided by Enron and $500 million by an outside investor, Whitewing Associates L.P. was formed. Two years later, the entity's arrangement was changed so that it would no longer be consolidated with Enron and be counted on the company's balance sheet. Whitewing was used to purchase Enron assets, including stakes in power plants, pipelines, stocks, and other investments. Between 1999 and 2001, Whitewing bought assets from Enron worth $2 billion, using Enron stock as collateral. Although the transactions were approved by the Enron board, the asset transfers were not true sales and should have been treated instead as loans.
- **Osprey**: In September 2000, a trust created by Enron in 1997 called the Osprey Trust, raised $2.4 billion from institutional investors in a so-called private placement of notes due in January 2003. Osprey was used as a central fund-raising entity for several of Enron's partnerships, including LJM and Whitewing.[2]

Data Set

The data set used for this predictive coding exercise is public domain data from Enron that originated from the Federal Energy Regulatory Commission (FERC) Enron Investigation. The data set consists of **92%** of Enron's staff emails, which originated in Lotus Notes (which Enron used as their email and work product platform). That data was reconstituted as Outlook PST files with attachments for the EDRM Data Set Project in 2011.

The PST collection of Enron data spans **171** PST files for **151** custodians, over **40** GB of data when unzipped. Including the PST container files, the collection contains **2,751,895** documents, of which **1,239,207** documents are emails. The rest are attachments in various file formats, including work product, media, system and other file types. This is the data set that is being used for this predictive coding exercise.

Goal of the Exercise

The goal of the exercise is to use Intella Connect's predictive coding module to identify documents associated with six Enron special purpose entity categories: **JEDI**, **Chewco**, **LJM**, **Raptor**, **Whitewing** and **Osprey**. This reflects a real-life use case that could have been applied during the Enron investigation as the special purpose entities were key to determining the extent of the accounting irregularities and fraud claims.

1. Source: Wikipedia (*https://en.wikipedia.org/wiki/Enron_scandal#Special_purpose_entities*)
2. Source: New York Times (*https://www.nytimes.com/2002/01/25/business/enron-s-collapse-mutual-funds-many-may-be-surprised-to-be-enron-investors.html*)

Identifying the Starting Candidate Data Set

One of the biggest areas of potential failure in predictive coding projects is failure to identify an appropriate starting candidate data set for use in the predictive coding exercise. Predictive coding algorithms are almost never applied to an entire document collection – there are several levels of culling that are typically applied to a data set before the actual predictive coding process begins. In this exercise, there were two main levels of culling to get down to the final set:

- **Retrieving and De-duping Set**: Performing a search to identify potentially responsive documents and deduplicating the results; and
- **Excluding Files Not Conducive to Predictive Coding**: Eliminating files from search results based on type or size of file that are typically not good for training or classification by predictive coding algorithms.

Retrieving and De-duping Set

In this exercise, we performed a multi-modal approach to predictive coding, beginning with the performance of a keyword search to identify potentially responsive documents to support the goal of the exercise. Based on an email containing document preservation instructions issued to all Enron employees on November 29, 2001, we selected the following subset of terms from that list to identify potentially responsive documents:

> ***ljm OR chewco OR jedi OR "joint energy development investments" OR Marlin OR whitewing OR raptor OR Andersen OR fastow OR dabhol OR Atlantic OR Osprey OR Braveheart OR Yosemite OR MEGS OR Margaux OR Backbone OR Nahanni OR Moose OR Fishtail OR Blackhawk OR CalPERS OR cuiaba OR Dynegy OR audit***

That search retrieved **114,317** files, as illustrated in Figure 1. Once deduplication was applied, the number of unique files responsive to the search was reduced to **27,969** files.[3]

Excluding Files Not Conducive to Predictive Coding

As mentioned above, the next step should be exclusion of all items, which don't have valuable textual content. This includes items which do not have textual content at all (for example: images, empty documents, archives, binary files, etc.) as well as documents that are not well suited for language processing (such as: spreadsheets, HTML or XML files, CSV files, source code, etc.) and documents that contain so much text that the text becomes a level of "noise" counter to effective classification. These documents should be suppressed from the Predictive Coding review.

Applying exclusion criteria within the Enron Data Set looks like figure 2:

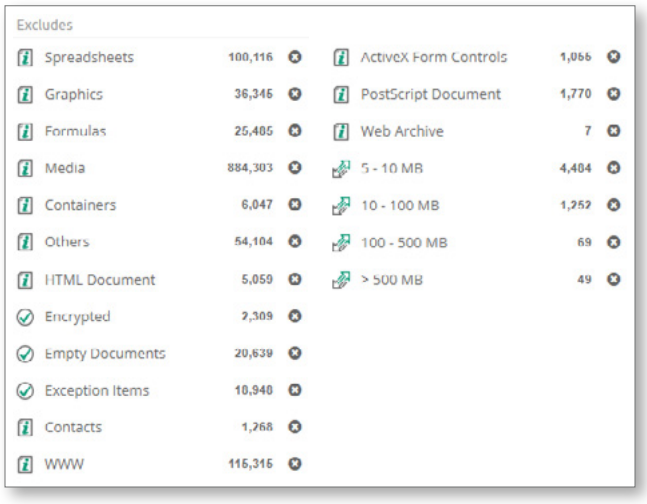

Figure 2: Excludes list (based on enron data set)

Excluding these document categories will help maximize the effectiveness of the algorithm in classifying documents based on reviewer training.

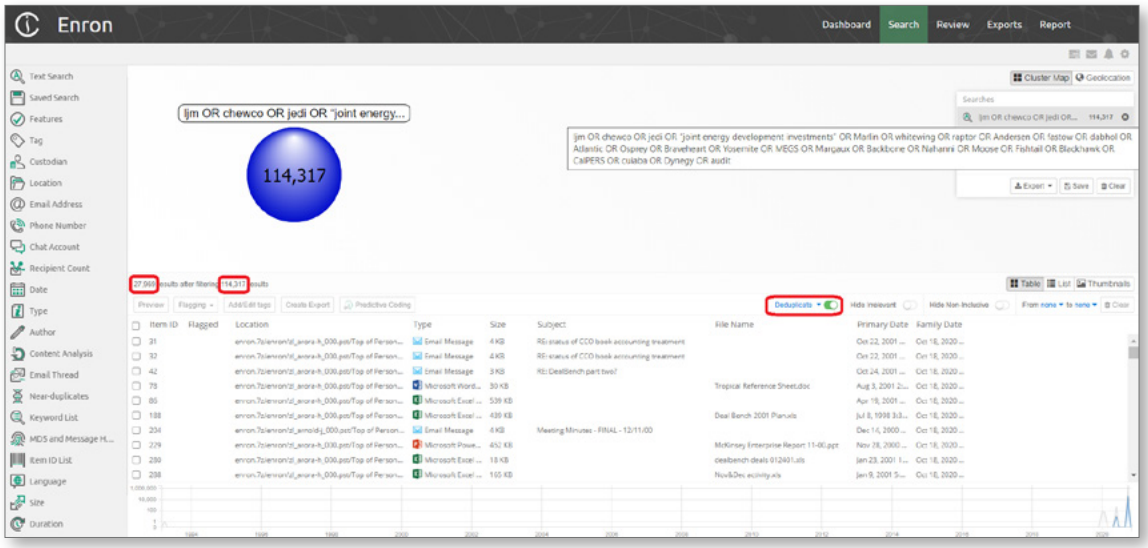

Figure 1: Search applied to enron data set

3. This level of deduplication within the Enron data set is common as many emails were sent to multiple parties.

Establishing the Final Predictive Coding Review Set

Once the excluded files are applied to the deduped result set, the result looks like figure 3:

Applying the exclusion set gives us a final total of **20,849** documents to be included in the predictive coding candidate set.

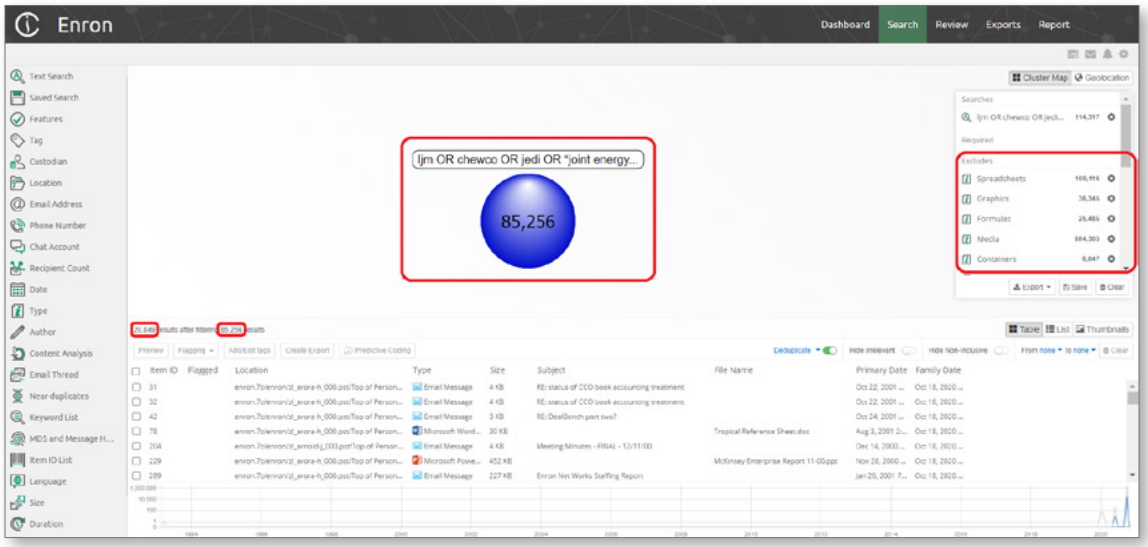

Figure 3: Result set after exclusions are applied

About Intella Connect Predictive Coding

Intella Connect utilizes a Continuous Active Learning® (CAL®) approach to predictive coding (also known as "TAR 2.0"). As defined by the *The Sedona Conference® Glossary, eDiscovery & Digital Information Management, Fifth Edition*, CAL is:

> *A machine-learning algorithm that periodically analyzes users' decisions in order to rank unreviewed data, with the most likely desired data ranking first based on the users' previous decisions.*

According to the *most recent Predictive Coding Technologies and Protocols semi-annual survey* conducted by Complex Discovery, Continuous Active Learning was reported as "the most used predictive coding protocol with **88.24%** of responders using it in their predictive coding efforts."

This section discusses the steps within the Intella Predictive Coding workflow. In the following section, we will apply those steps to the predictive coding candidate set identified above.

Overall Process

The overall predictive coding process can be illustrated by Figure 4 opposite.

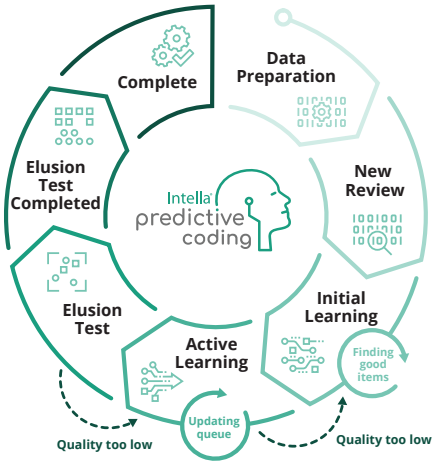

Figure 4: Diagram of the overall process using intella connect

Data Preparation

When a predictive coding candidate set is ready for review, the first step is to create the review within the predictive coding module. The parameters that are set for this stage impact the review to follow. They are, as follows:

- **Name**: the unique name this Predictive Coding review will be identified with
- **Coding Layout**: the name of the coding layout you would like to be displayed in the review console.
- **Category field**: the engine will use that field to categorized items in your review. It should have only two possible options. Available fields will be populated with using coding layout selected previously.

- **Primary option**: option selected here will be considered as the category of your prime interest (i.e,. items of your interest; usually "Relevant"). It is important to select the right option here, as the entire review will be optimized to find items later coded with this option.

You can also configure the engine using following advanced settings:

Algorithm: allows to select one of two available algorithms. One is considered faster, but less precise. Reviews using the other one may have better quality, but they are three times as expensive to prepare and evaluate. You can read more about those in later chapters.

Max CPU cores: defines a limit of how many processor cores may be used during learning of the model.

Use existing tags to teach the model: if you leave this option selected, then the engine will use existing tags assigned to items to train the model, skipping the "Initial learning" and going straight to the "Active learning" phase. Using this option with a handful of correctly coded items can expedite the learning process, especially if certain highly relevant documents are already known.

Figure 5 is an example of the form that is used to kick off data preparation:

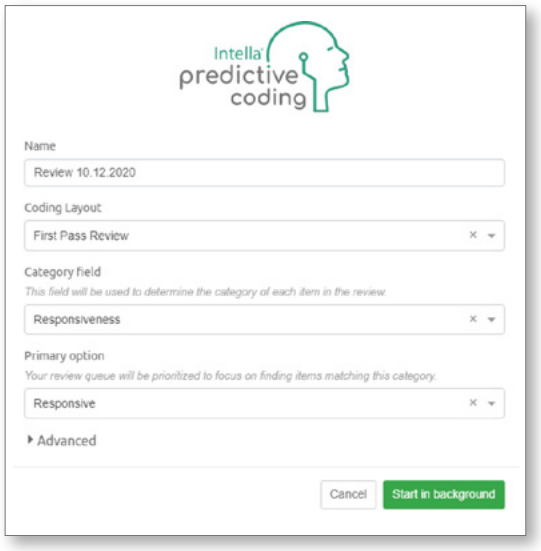

Figure 5: Kicking off data preparation

Once you click on the button "Start in background", expect the data preparation to take some time. The process could take several hours or more, so it's best to launch it at the end of the work day to give it time to hopefully prepare the data set by the next morning.

New Review

Once data pre-processing has ended, Intella Connect will create a new, blank model that will handle your review. Assuming that you haven't chosen to use existing tags to teach the model, the model initially does not have any preconceptions about what items to look for. However, the pre-processing has already determined measures for the textual similarities and differences between these items.

This is symbolically presented in figure 6. Dots represent individual items. For sake of simplicity, they are placed in a two-dimensional space, with similar items being placed near each other. You can easily observe that certain groups are formed: some items cluster together, others are spread further apart. The model itself is represented by the rounded square in the middle.

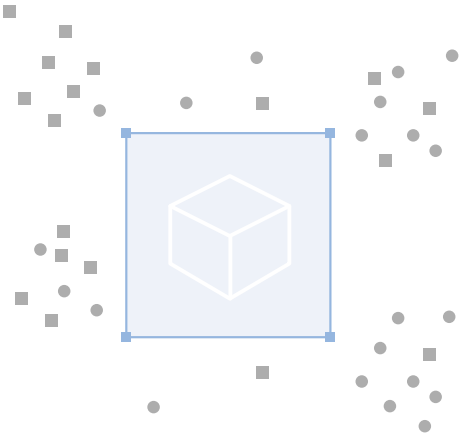

Figure 6: Representation of data set after pre-processing

At this point, the items in the review queue are still in random order. The model will start changing its state once the very first item gets coded by a reviewer.

Initial Learning

Assuming that you haven't chosen to use existing tags to teach the model, the next step will be Initial Learning. If you have chosen to use existing tags, this step will be skipped and the workflow will move straight to Active Learning. For Initial Learning, as soon as the first item in the review queue is coded, the model starts moving randomly through the items space, giving you items that are significantly different from one another, while trying to focus its search around items that you code as relevant.

Figure 7 illustrates how items from different groups are being attracted to the model. When you make a coding decision, those items will stay in the model and be used for the learning process.

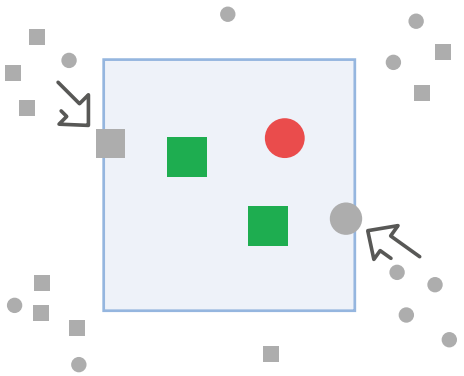

Figure 7: Representation of initial learning being used to begin to train the algorithm

It is perfectly acceptable to see many non-relevant items at this point, as it highly depends on the ratio of relevant items in your case. However, there is already much more going on than just selecting random items for review. You get the benefits of artificial intelligence soon after the very first item has been reviewed.

After you have coded the 10th item, the first iteration of the learning process will take place. The model will self-assess if it already can give you any items that are likely to be relevant. If it can - it will move to the next stage of the review. If not, then it may remain in "Initial learning" state for several iterations.

Active Learning

In this stage, the model evaluates all items in the review queue and will score them individually according to the knowledge inferred from previous coding decisions (either from Initial Learning or existing tags). Then, it sorts those items in descending order of predicted relevance, putting the ones scored the highest at the front of the queue.

The model then delivers items from the front of the queue for you to code. Those have the highest probability of being relevant, so you should now start to see the value of using the Predictive Coding engine. Instead of reviewing random items, your queue is now optimized to find the most relevant items first.

Keep in mind that you can still see non-relevant items at the front of the queue. This is normal, as this is an iterative process and it may take several iterations until the model knows how to best separate relevant from non-relevant items. The more iterations you go through, the better it the model will get at making those determinations.

After a certain number of items have been coded, the next iteration will start and the model will once again update your review queue. Over time you should see that the gap between the scores of the relevant and non-relevant items grows larger with each iteration. This is an indication that the model is getting better at separating these two categories, as seen in Figure 8.

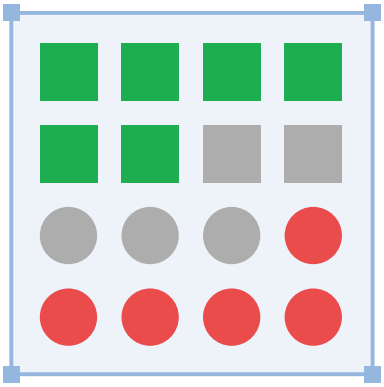

Figure 8: Representation of data set as active learning continues to train the model

If at some point in time the model goes back to the "Initial learning" stage, that is an indication that it cannot properly distinguish between relevant and non-relevant items based on the human coding decisions that it has analyzed.

There is never a clear sign as to when to stop the review when it is in the learning stage, but there are a few indications when it may be wise to do it:

- The model starts to present less and less relevant items in each iteration.
- The gap in the scores assigned by the model is large (ex. the score of the last relevant item is 70% and for the first non-relevant item is 20%)
- You've achieved the desired recall level (Elusion test can be used for this)
- You've met other metrics set for your project, e.g. allowed time.

When case manager decides that the time is right, he should move to the next stage of the process, which is the Elusion Test.

Elusion Test

An Elusion Test is a process in which you verify how many relevant items are still left in the previously unseen (not coded) items space. This is illustrated in figure 9.

In this example our model has already refined its predictions about the category of each item, but there are typically a large portion of the items yet not yet reviewed (represented by gray dots).

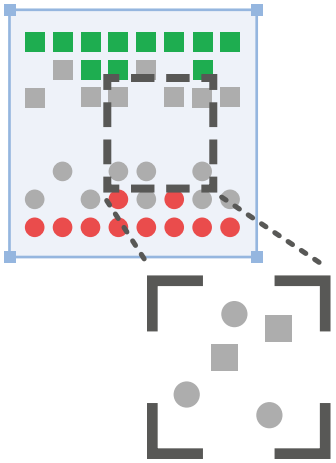

Figure 9: Selection of random sample of uncoded items for elusion test

The Elusion test will take **a random sample** of the non-coded items for you to review using the standard review UI, providing the option to select a fixed size of the document collection or the number of documents necessary to achieve a certain desired recall percentage. Once it's complete, it will measure and estimate how many relevant items could potentially be left in the set of non-coded items. Based on this statistic, you can opt to either continue the review (go back to active learning) or accept the model.

An Elusion test protects you against blind spots in the model. A low amount of relevant items in the last iterations may simply be due to the Predictive Coding not having seen a particular subtype of relevant documents yet. The Elusion test will help ensure that these are not overlooked.

Elusion Test Completed

Once all items in the elusion test have been coded, you will be presented with statistics informing you about the outcome of the test. The perfect situation for a perfect Predictive Coding model is obviously to find zero relevant items in the test, but this is rarely the case.

Typically, a portion of test items will be relevant in every Elusion Test review. Based on this value we can calculate a point estimate of eluded items in the non-coded part of the items space. We also calculate a min-max range of eluded items, based on a confidence level set at 95%, which is a typical confidence level value used in statistical analysis for purposes like these.

You will also get an estimated recall for your entire review queue.

At the end of this stage, you will be prompted to make a decision about what to do with your Predictive Coding model

- Continue review
- Accept the model

If you opt to continue the review, we will take the items coded in the elusion test and combine that knowledge with the model to learn new information about your items. A next iteration of learning will occur and you will go back to the "Active learning" stage.

It may be appropriate to run several elusion tests this way, until you get the level of quality that meets your requirements.

If you decide to accept the model, it will be persisted and fixated. At this point, you can no longer make coding changes through the predictive coding review console. This also means that the model will not be able to learn anything new.

Model Accepted

Once the model represented in figure 10 is accepted and its state cannot be changed, another option in the UI will unlock – you will be able to apply the model's predictions to the remainder of the review queue.

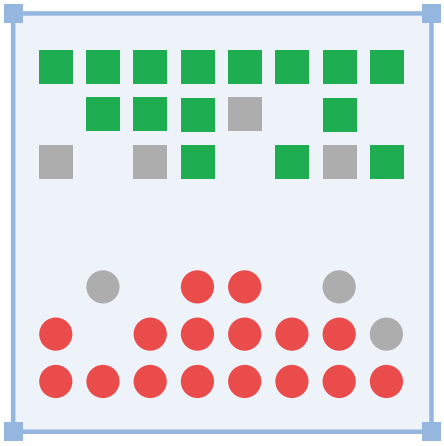

Figure 10: Representation of data set when ready to accept model

Applying the model to non-coded items is straightforward. Since the model can provide the relevancy score of each item in that pile, it's important to decide what the threshold value is above which we qualify everything as relevant, and below which items are qualified as non-relevant. The UI built for this will help you to select the right value and explain the resulting item counts.

Once that process is finished, all categorizations are applied, and the review is completed.

Completed

When the review workflow reaches this state, the model has gone through all the stages of development. The quality of the model has been accepted and its predictions have been applied to the non-coded items.

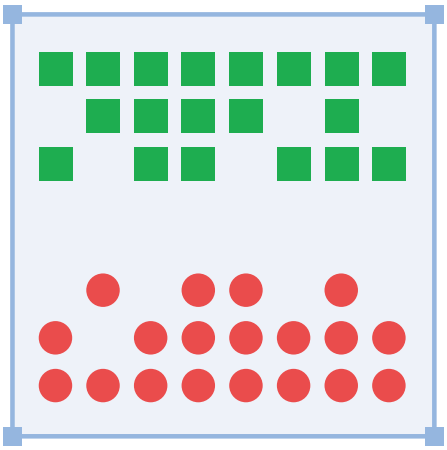

Figure 11: Representation of data set when model is completed

At this stage, the model is complete, as represented in figure 11. The workflow can proceed to post-review tasks for the documents determined to be relevant.

Predictive Coding Exercise with Candidate Data Set

Having discussed the creation of the predictive coding candidate set and the predictive coding workflow used by Intella Connect to conduct predictive coding on a data set, this section discusses how that workflow was applied to the predictive coding candidate set that was identified above for our exercise.

Using Existing Tags to Teach the Model

Before the predictive coding process was initiated, several documents were reviewed to identify potentially responsive and non-responsive documents to teach the model. This was done to ensure that certain documents identified as highly responsive would be used to "jump start" the training process. As a result, our exercise bypassed the Initial Review stage of the predictive coding workflow.

For this exercise, **43** total documents were reviewed before the predictive coding workflow was initiated. **15** documents were classified as responsive and **28** were classified as non-responsive. These "existing tags" were then used to teach the model at the start of Active Review.

Data Preparation

The parameters chosen to initiate data preparation are represented in figure 12:

Once the data preparation step was completed, the candidate set was ready for Active Learning.

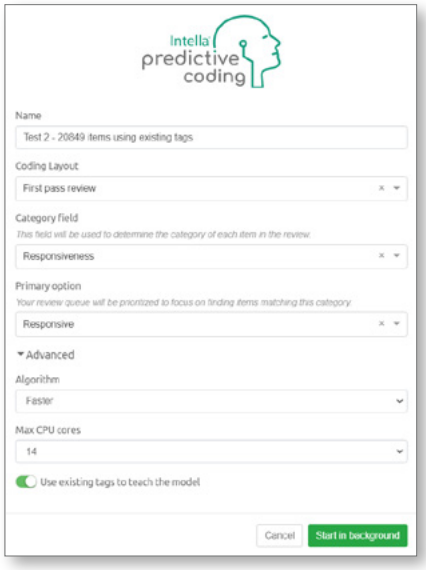

Figure 12: Data preparation parameters for Enron predictive coding exercise

Active Learning Review

As discussed in the ***About Intella Connect Predictive Coding*** section above, the predictive coding interface delivers documents in a prioritized order based on learning up to that point to deliver documents most likely to be responsive, enabling the reviewer to continue to train the algorithm as review continues by classifying documents as responsive or non-responsive. In addition, this review was set up using a template that also enabled reviewers to identify "hot" and privileged documents during that review, as shown in figure 13.

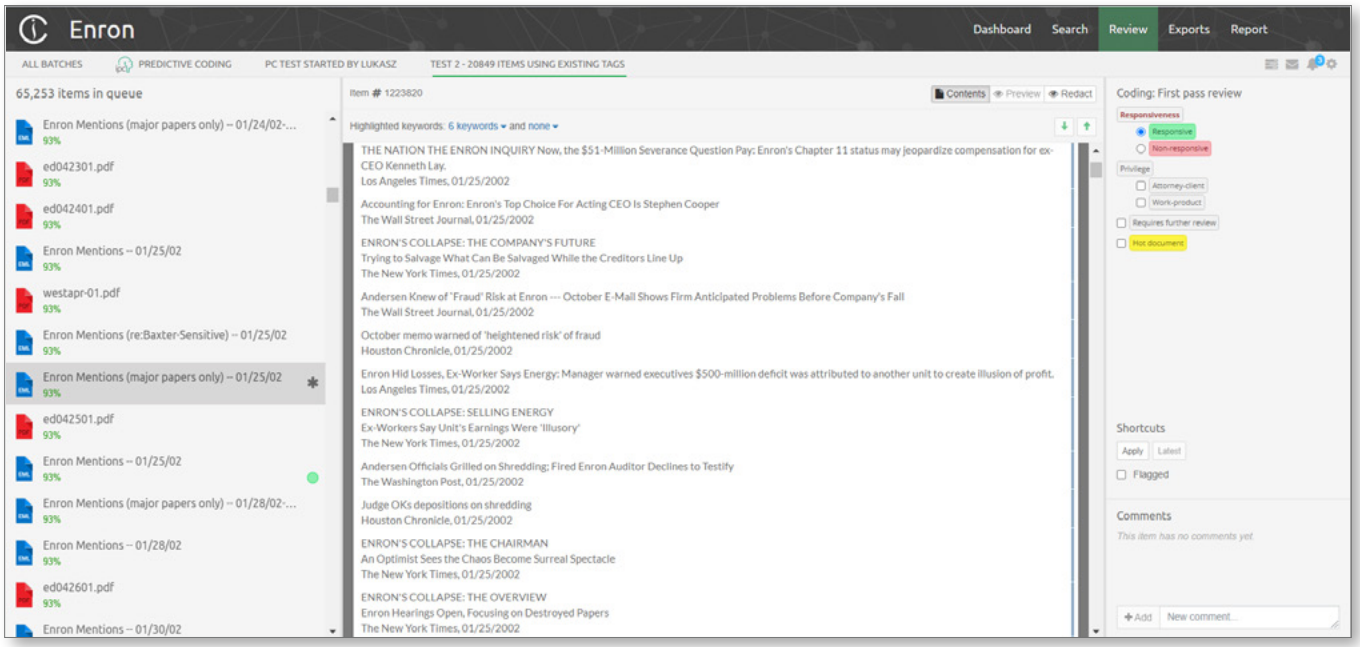

Figure 13: Active learning coding form for Enron predictive coding exercise

Active Learning Iterations 1 Through 19

As Active Learning began, most of the documents delivered by the algorithm in each iteration were responsive as seen in figure 14. In this exercise, only one of the first nineteen iterations had more non-responsive documents delivered than responsive documents. In many of the iterations, more than **90 percent** of the documents were responsive, so the algorithm was certainly prioritizing delivery of responsive documents well within the first nineteen iterations.

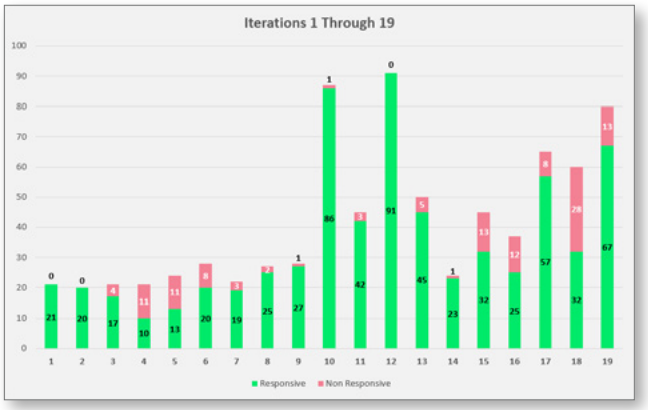

Figure 14: Active learning iterations 1 through 19 with mostly responsive documents

Active Learning Iterations 20 Through 26

During this phase of Active Learning, the model began to deliver more non-responsive documents as it began to run out of responsive documents at the top of the queue. As seen in figure 15, only three of the seven iterations here had more responsive documents than non-responsive documents – in those instances, continuous training found isolated areas of responsive documents not previously identified.

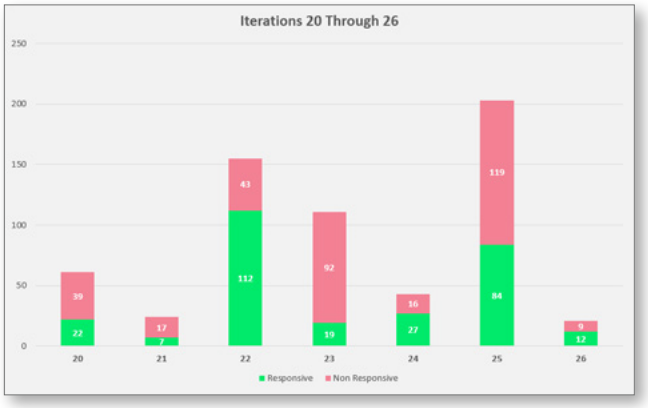

Figure 15: Active learning iterations 20 through 26 with a mix of documents

Active Learning Iterations 27 Through 32

By this point, the algorithm was running out of responsive documents to prioritize in the queue – to the point that, by the 32nd iteration, only **9.51%** of the documents were responsive.

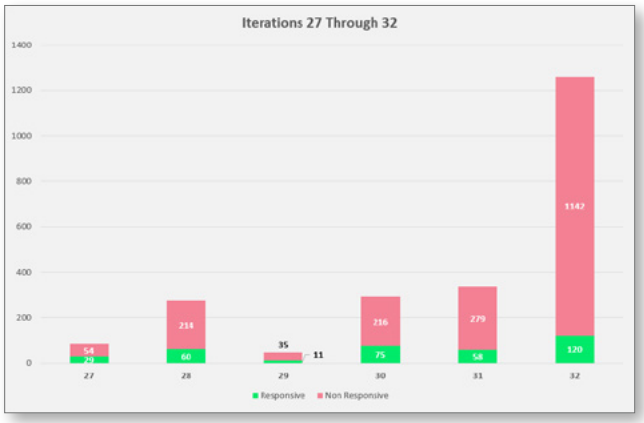

Figure 16: Active learning iterations 27 through 32 with mostly non-responsive documents

Figure 17 shows the ratio of responsive documents illustrates how the percentage of responsive documents eventually declined as the iterations began to run out of responsive documents to prioritize during review.

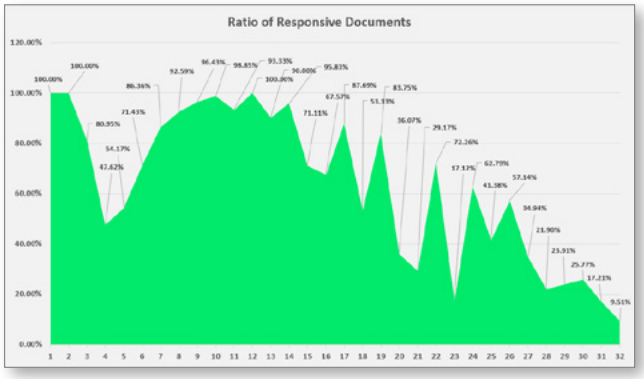

Figure 17: Ratio of responsive documents for all 32 active learning iterations

At this point, we determined we were ready to run our first elusion test to test the results.

Elusion Test

Proceeding with the Elusion Test, we have chosen to achieve a desired recall level of **75%**. As a result, the predictive coding interface has determined that we must review **4,579** items to match that statistical criteria and that the maximum eluded items (i.e., responsive documents not previously delivered by the algorithm) allowed to be identified is **145**.

We could have chosen a fixed size of documents to review here, but we chose instead to target a desired recall rate to provide verification for the model. This level of testing would help with defensibility if being asked to authenticate the results in a real world scenario.

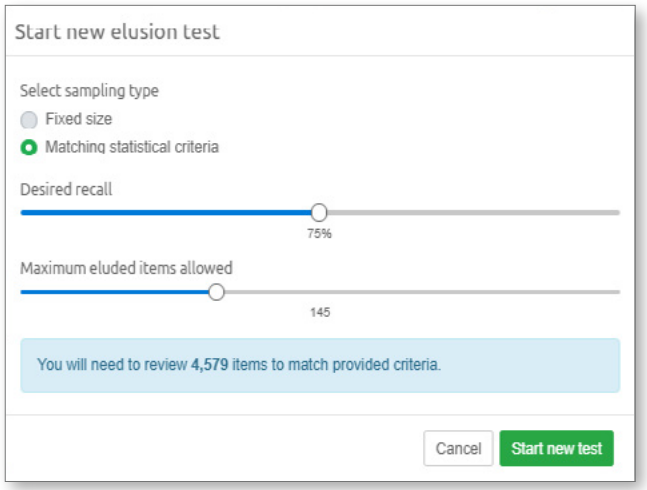

Figure 18: Form to start new elusion test for Enron predictive coding exercise

Once we selected Start New Test shown in figure 18, we proceeded to review the **4,579** randomly selected documents to test the results of our training.

Completion of Elusion Test

When the Elusion Test review was completed, the results looked like figure 19:

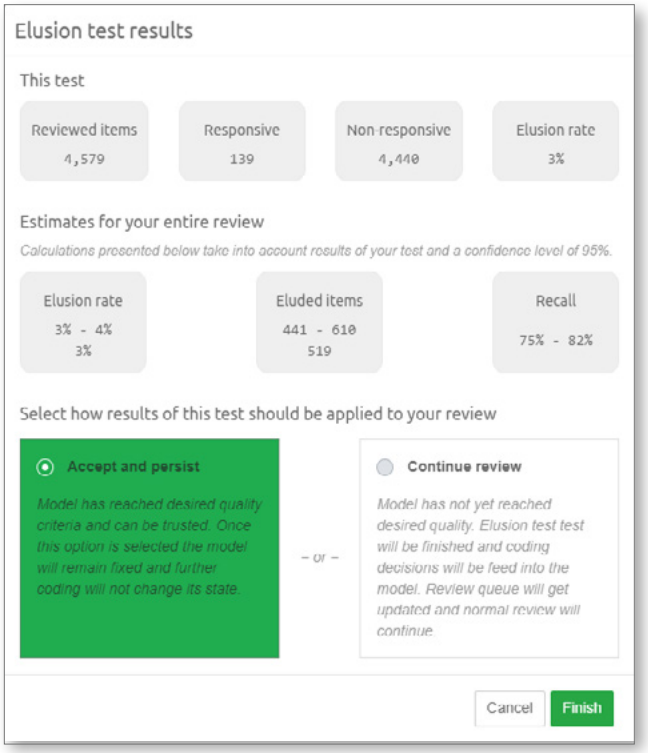

Figure 19: Elusion test results for Enron predictive coding exercise

As you can see, our review of the Elusion Test set identified **139** responsive documents, so the number of documents that eluded identification by the predictive coding module was under the threshold of **145** documents that was identified at the start of the Elusion Test. Our projected recall rate is also well within the goal of **75% recall**, so we can accept the model and persist to lock in the review we have performed and apply coding to the remaining items not reviewed. Had we found more eluded items than the threshold, we would have decided to continue review and training with additional Active Learning iterations – since we didn't, we're ready for the final step to apply classifications to the remainder of the collection.

Apply to Remaining Items

When we selected Apply to Items for the predictive coding review exercise, the predictive coding interface displayed the following form for us to set the threshold level for coding the documents. As you can see by the form below, only **289** documents have a current score of higher than 25% likely responsive, which is only **2.31%** of the remaining non-code collection. Applying coding to the remaining items will code **289** documents as responsive and **12,231** documents as non-responsive. This will complete the coding review for our Enron predictive coding exercise.

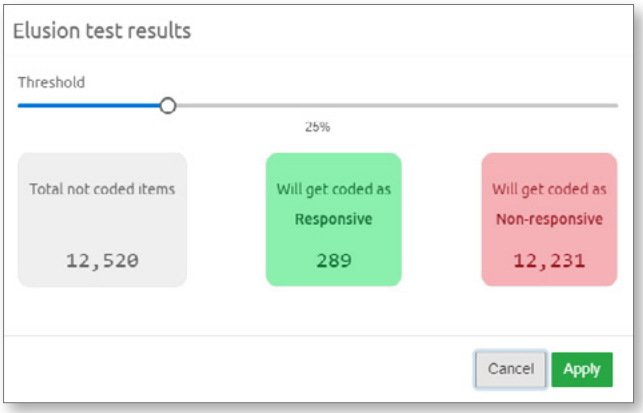

Figure 20: Applying coding to remaining results with Enron predictive coding exercise

We can now look at the final results, of our predictive coding exercise in Figure 20, in terms of how many documents were reviewed vs. documents we didn't have to review, how many documents were classified as responsive vs. non-responsive, and potential savings in time and effort from our predictive coding exercise.

Final Analysis

The breakdown of documents classified is illustrated in the Categories Distribution graph in figure 21:

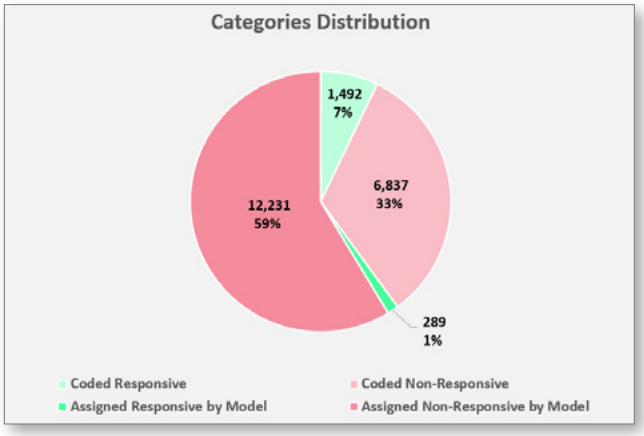

Figure 21: Distribution of documents in Enron predictive coding exercise

If this were a live case, our team could then prepare to produce the total number of documents identified as responsive (**1,492** + **289** = **1,781** total responsive documents) plus any family members of those documents as the complete production.

Conclusion

While we did review just under 40 percent of the collection (**39.9%**, **8,329** total documents reviewed), we avoided reviewing more than 60 percent of the collection (**60.1%**, **12,520** total documents classified by the model). Assuming a reasonable rate of **40 documents reviewed per hour** (equivalent to 1.5 documents per minute) to review all documents and a reviewer billing rate of **$50 per hour** (for typical document review attorneys), the time and cost savings could have been as follows[4]:

Total time saved: **12,520** documents / **40** documents per hour = **313** hours of review saved

Total cost saved: **313** hours of review saved x **$50** per hour review rate = **$15,650** total review cost savings

This illustrates potential savings in time and cost for review of a data set of this size. As you can imagine, the potential savings of both could even be greater for larger data sets to be reviewed. The end result of our predictive coding review exercise for this exercise using the Enron data set is a review process that saved thousands of dollars in review costs while also being verified as an accurate review result.

4. Review throughput rates and billing rates vary widely, so this is just an example of the potential savings.

PHONE ENQUIRIES
+1 (888) 291-7201

AMERICA america@vound-software.com
ASIA asia@vound-software.com
EUROPE europe@vound-software.com

www.vound-software.com

Vound
AMERICA • ASIA • EUROPE