Columbia Business School
AT THE VERY CENTER OF BUSINESS™

Columbia CaseWorks

ID#190202

PUBLISHED ON
MAY 3, 2019

# An Introduction to AI for Text Mining: A Companion to the Evisort Case[i]

BY C. DANIEL GUETTA*

## Introduction

The Evisort case discussed the story of Evisort—an AI-powered start-up founded by Amine Anoun, Jake Sussman, and Jerry Ting in 2013. Contracts are the scaffolding of the modern company, with the data contained in them supporting many critical decisions for the whole organization, from operations, to supply chain, to advertising. Evisort uses AI-driven text mining, a relatively new business analytics tool, to unlock the veritable troves of information contained in these contracts and make them searchable by content and contract metadata.

Evisort's ability to analyze contracts in such a fashion is a testament to the recent progress in the fields of machine learning, artificial intelligence, and business analytics. Indeed, the process of analyzing a contract and extracting the metadata therein comprises a number of intricate steps. First, optical character recognition must be used to convert a contract image into machine-readable text. The text must then be converted to a format in which it can be analyzed (algorithms work on numbers, not text). Finally, information must be extracted from the contracts, whether it be about the contract as a whole (e.g., the contract type) or about specific clauses therein (e.g., the counterparties or relevant dates).

In this note, we shall delve into some of the details involved in carrying out these steps.

The importance of text mining goes far beyond the confines of this particular case. Text is everywhere. To name a few examples:

- Medical records and doctors' notes



[i] This industry note is written as a companion to Columbia CaseWorks case ID#190201A, "Evisort: An AI-Powered Start-up Uses Text Mining to Become Google for Contracts," by C. Daniel Guetta, November 9, 2018. It can, however, also be used as a stand-alone introduction to AI for text mining.

**Author affiliation**
*Columbia Business School*

- Complaint records
- Inspection or repair notes
- User-generated content, including social media posts and product/service reviews
- Financial reports
- News articles

Mastering the analysis of text data can therefore unlock the information contained in these data sources. We will be illustrating the techniques of text mining in the context of Evisort, but these could translate to any of the applications above.

## The Use Cases

We will anchor our discussion in two very different organizations,[ii] each seeking to use Evisort for different purposes—a small credit union in the Pacific Northwest, and an inventor's cooperative in Brooklyn.

### THE PACIFIC TRUST CREDIT UNION

The Pacific Trust Credit Union was originally created to serve gold miners who flooded the area in the late 1800s and has evolved to serve freelancers and gig-economy workers, who now make up an estimated 70% of its customer base. The PTCU provides a number of essential banking services, together with small business and personal loans. Remarkably, the governance structure of the credit union has stayed mostly unchanged throughout its long history. Driven by the idealism of its founders and successive leaders, it has operated with minimal staff, and a very informal management style.

Unfortunately, the credit union's fortunes have started to shift. Lower interest rates have put pressure on its profits, and the tightening competition in the gig economy has increased pressure on its customers. In addition, even the PTCU's most loyal customers are starting to feel the impact of the credit union's antiquated systems and are requesting access to more modern services, such as online banking. As a result, the PTCU's leadership has decided the time may have come to merge with a larger entity.

Every potential partner it reached out to balked when faced with the consequences of more than 100 years of substandard paperwork and operational controls. Potential partners had two main concerns:

1. **Issuance of loans**: The credit union used a number of experienced loan officers, all of whom were trusted to make loan decisions on its behalf. Although default rates were far below the industry average, this decentralized plan resulted in inconsistent loan contracts with different wording, time lines, and terms.

[ii] The two use cases are fictitious institutions; all data, persons, and events are composites based on common business scenarios.

Placing trust in each of the loan officers to manage the collection of their own loans, the credit union did not have any centralized collection-tracking system.

2. **Compensation decisions**: The way PTCU handled compensation of its loan officers and its small cadre of administrative staff was also ad hoc. Contracts were renegotiated with the elected chair of the credit union at the time, with little consistency in terms or salaries.

Given these concerns, no larger company was willing to partner with the PTCU. The leadership had to urgently address these concerns. Luckily, the organization had scanned its contracts since the advent of the technology, so they were available digitally, but it had done nothing to classify or annotate them. PTCU turned to Evisort for urgent help. To start, it asked Evisort to compile and analyze all the credit union's contracts, and identify loan agreements and compensation contracts for further analysis.

## THE INVENTOR'S DEN

The Inventor's Den was launched in early 2002 in downtown Brooklyn to serve as an umbrella company for entrepreneurs of every stripe. From novelty chefs, to website designers, to day traders, entrepreneurs could use the Inventor's Den as a corporate entity from which to run their businesses.

As the company increased in size, its range of activities grew, and some of these activities started attracting the attention of the authorities. In particular, two of the Den's entrepreneurs were focused on marijuana and blockchain, respectively.

The resulting audit quickly revealed a major issue with the Den's operations. Each entrepreneur was able to enter into contracts independently on behalf of the Den. All they had to do was send a copy back to headquarters and get a rubber-stamp signature. As a result, the Den had entered into a large number of contracts, with little oversight or input from its management team.

With Evisort's help, the Den sought to evaluate all these contracts and group them into a number of coherent categories. The Den would then be able to hire experts to further investigate the issues raised, by category.

It is important to note that this particular problem we have chosen—contract classification into categories—is crucial to Evisort's value proposition. As we saw in the first part of the case, one of Evisort's key value propositions is its ability to provide value to all parts of the company, not just to its lawyers. For example, a company's supply chain and procurement team might want to use its procurement contracts to help optimize supply-and-demand decisions. Or a newly hired head of real estate might need to find all the company's lease contracts to gain a better understanding of the company's real estate obligations. To meet these aims, it is essential for Evisort to give the companies the capability to automatically determine the category of a contract without having to read through it, so that they can pull up all relevant contracts quickly.

Finally, we highlight one key difference between these two cases—in the first, the PTCU *knows* the categories it seeks to find among its contracts, and can presumably use past examples of such contracts to create models. In the second, the Inventor's Den is looking to find categories without any past examples.

## The Data

To illustrate these problems, we shall use a publicly available data set that closely mirrors the kind of information Evisort would have access to when starting with a new client. This data set is gathered from Securities and Exchange Commission public filings. Whenever a publicly listed company signs what is called a "material contract" (i.e., a relevant contract that could affect the company in a significant way), it must file a copy of this contract with the SEC, and the agency then makes these contracts publicly available. We rely on a company called Law Insider, which aggregates these contracts and classifies them by category. (For example, Law Insider's categories include "lease" and "employment agreement.") As an initial step, we condense some of these categories, sanitize the data, and provide the companies' contracts in two files:

- The file contracts.txt, in which each line contains the full text of one contract. The file contains 1,000 contracts.
- The file topics.txt, which lists the category of each contract in contracts.txt. Each line in this file corresponds to the contract in the same line number in the contracts file. The categories are:
    - loan
    - lease
    - credit
    - stock purchase
    - incorporation
    - employment
    - compensation
    - purchase agreement
    - consulting agreement
    - merger



## Why Text Is So Difficult: The Challenge of Unstructured Data

Before we look at our use cases in detail, it is helpful to understand why text data are so difficult to deal with. To understand this, we distinguish between *structured* and *unstructured* data. Structured data are what we might typically refer to as "data"—tabular records in which each row corresponds to an entity, and each column lists a particular aspect of the entity—usually in a numeric or well-defined format (such as a date, or one of a few allowable values, such as "male," "female," or "other"). Unstructured data lack this structure—they might

contain the same information and more, but in a form that cannot readily be understood by a computer.

To illustrate the distinction more concretely, consider figures 1a and 1b below. Both figures list information about the same (fictitious) contracts, but the data in figure 1a are unstructured, whereas those in figure 1b are structured.

**Figure 1a:** contracts represented as unstructured, raw text data

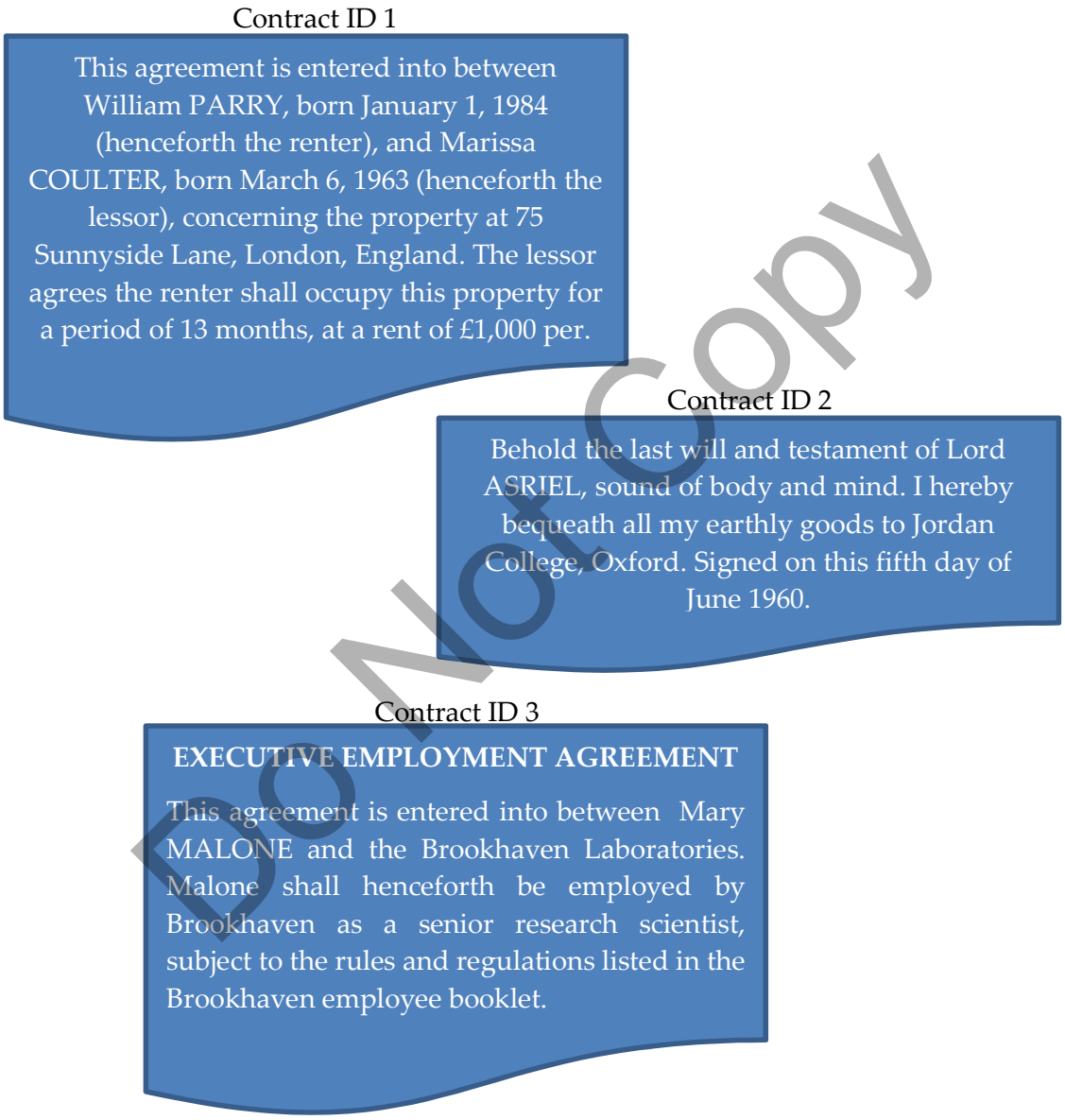


**Figure 1b:** structured data extracted from the contracts in figure 1a

| Contract ID | Contract type | Party 1 | Party 2 | Party 1 DOB | Party 2 DOB | Signature Date |
|---|---|---|---|---|---|---|
| 1 | Lease | William Parry | Marissa Coulter | Jan 1, 1984 | Mar 6, 1963 | |
| 2 | Will | Lord Asriel | | | | Jun 5, 1960 |
| 3 | Employment agreement | Mary Malone | Brookhaven Laboratories | | | |

In this particular case, we arbitrarily chose seven fields of interest to store in our structured data, but there are others we left out (for example, the terms of the contract, or the aliases by which the counterparties are referred to in the contract). It is difficult to devise a structured format that could capture every conceivable relevant piece of information.

Jake Sussman, chief operating officer of Evisort, often came across companies that had attempted to create structured indexes for their contract databases, such as that illustrated in figure 1b. According to Sussman, these databases often failed when a new contract contained information the company had not included in this structured format, or, more frustratingly, when the company realized it needed to query data it had neglected to capture in a structured form.

Evisort's approach is to extract relevant data *directly* from the contracts themselves. This, of course, is a difficult task. Text data are relatively dirty—words can be misspelled, people can write ungrammatically, abbreviate unpredictably, use synonyms, and use domain-specific terminology. To make matters even more complicated, context is far more important in text data than in structured data. The essence of text mining is to extract these structured pieces of information from the unstructured text, in spite of these difficulties.

Although this note will focus on text mining for contracts, the structured/unstructured distinction is not limited to contracts, or even to text data. Images and sounds are another common source of unstructured data. Similar processes to the ones described in this case exist for these other types of unstructured data.



## Structure from Chaos

The first step will be to represent the text in a form that is amenable to analysis. AI algorithms work with numbers, not with text—so we need to convert the text to numerical data. In general, the process of representing or *encoding* text in this fashion comes at the expense of losing some of the information in the original text data. The more complex the encoding method, the more of the initial information is kept.

It should be noted that the representation of text data is in many ways as much an art as a science. We introduce a number of tools in this section, but they are by no means the only

possibilities, nor are they all necessarily appropriate in every situation. Considerable experience is required to determine the subset of text representation techniques that are appropriate for a given situation.

### THE BAG-OF-WORDS REPRESENTATION

The most basic method used to encode text is called the *bag-of-words* representation. It views a piece of text as a sequence of words, and ignores word order, punctuation, formatting, capitalization, and characters other than letters. The approach simply counts the number of times each word appears in each document.

Figure 2 shows the start of the bag-of-words representation of the contracts in figure 1a.

**Figure 2a**: the bag-of-words representation of the contracts in figure 1a

| | Contract ID 1 | Contract ID 2 | Contract ID 3 |
|---|---|---|---|
| a | 1 | 0 | 1 |
| agreement | 1 | 0 | 2 |
| agrees | 1 | 0 | 0 |
| all | 0 | 1 | 0 |
| and | 1 | 2 | 2 |
| as | 0 | 0 | 1 |
| at | 2 | 0 | 0 |
| asriel | 0 | 1 | 0 |
| behold | 0 | 1 | 0 |
| bequeath | 0 | 1 | 0 |
| between | 1 | 0 | 1 |
| body | 0 | 1 | 0 |
| booklet | 0 | 0 | 1 |
| born | 2 | 0 | 0 |
| brookhaven | 0 | 0 | 1 |
| … | … | … | … |



Representing text in this form clearly results in the loss of a significant amount of information. For one, word order is completely lost. The bag-of-words representations of "David shall borrow $20 from Ben" and "Ben shall borrow $20 from David" are identical, though the sentences differ in a key way. Punctuation is also lost—the bag-of-words representations of "my panda eats, shoots, and leaves" and "my panda eats shoots and leaves" is also identical, despite the fact that the former describes a considerably more violent pet. Finally, capitalization is also ignored—thus, the bag-of-words representations of "Hitchcock shot The Birds in 1962" and "Hitchcock shot the birds in 1962" are identical, despite the fact that the former refers to moviemaking, while the latter chronicles a hunting vacation.

Despite this loss of information, the bag-of-words representation can lead to some powerful insights, as we shall later see. The remaining methods we discuss all build on the bag-of-words representation.

### STOP WORDS, STEMMING, AND LIMITING FREQUENCIES

In addition to the loss of information it entails, the bag-of-words approach suffers from other shortcomings. Among them are the following:

- It gives equal weight to words with little semantic meaning (e.g., "a," "and," "the") and words laden with meaning (e.g., "lease," "bequeath").
- It does not account for the relationship between words that derive from each other. For example, the words "eat," "eats," "eating," and "ate" would all be considered completely distinct and separate words by the bag-of-words representation, even though they are more similar to one another than to "house," for example.
- It results in very large representations of the text data, sometimes far larger than the initial corpus. In the bag-of-words representation, one row is needed for every word in the vocabulary, even if it appears in only one or two documents. For example, the SEC contracts provided with this case contain over 100,000 distinct words. Thus, the bag-of-words representation would be a table with over 100,000 rows. This can make it more difficult to build models using these features, as we later discuss.

Refinements to the bag-of-words method address each of these problems.

First, we discuss the concept of *stop words*. "Stop words" refer to the most common words in a language, which are likely to carry little meaning. In English, these are words such as "a" and "and." It is common to exclude these words from text before the bag-of-words representation is created. Figure 2b shows the bag-of-words representation in figure 2 with common stop words removed.

**Figure 2b**: the bag-of-words representation in figure 2, with common stop words removed

| | Contract ID 1 | Contract ID 2 | Contract ID 3 |
|---|---|---|---|
| agreement | 1 | 0 | 2 |
| agrees | 1 | 0 | 0 |
| asriel | 0 | 1 | 0 |
| behold | 0 | 1 | 0 |
| bequeath | 0 | 1 | 0 |
| between | 1 | 0 | 1 |
| body | 0 | 1 | 0 |
| booklet | 0 | 0 | 1 |
| born | 2 | 0 | 0 |
| brookhaven | 0 | 0 | 1 |
| … | … | … | … |

Of course, stop words vary by language. (For example, the French word for "all"—"*tout*"—would likely be considered a stop word in French. In English, on the other hand, "tout" would certainly not be a stop word.)

Next, we discuss *stemming*. Stemming describes the process of reducing inflected and derived words to their word stem. For example, reducing "agreement" and "agrees" to "agree." Stemming is a surprisingly tricky process, and heavily language-dependent. Fortunately, stemming is studied in considerable depth by linguists, and a number of efficient stemming algorithms are available for use in many common programming languages. Perhaps the most widespread of those is the Porter stemming algorithm. Stemming algorithms are applied to the text before word counts are calculated. Figure 2c shows the bag-of-words representation in figure 2 after stemming.

**Figure 2c**: the bag-of-words representation in figure 2 after stemming

| | Contract ID 1 | Contract ID 2 | Contract ID 3 |
|---|---|---|---|
| a | 1 | 0 | 1 |
| agree | 2 | 0 | 2 |
| all | 0 | 1 | 0 |
| and | 1 | 2 | 2 |
| as | 0 | 0 | 1 |
| at | 2 | 0 | 0 |
| asriel | 0 | 1 | 0 |
| behold | 0 | 1 | 0 |
| bequeath | 0 | 1 | 0 |
| between | 1 | 0 | 1 |
| body | 0 | 1 | 0 |
| booklet | 0 | 0 | 1 |
| born | 2 | 0 | 0 |
| brookhaven | 0 | 0 | 1 |
| … | … | … | … |



A final technique we shall discuss in this section is the use of *limiting frequencies*. Stop words and stemming can reduce the number of words needed in a bag-of-words representation, and can eliminate many words without semantic meaning. Nevertheless, many words without much predictive power can still remain in the representation. To remove those, we begin by noting that words that appear in most of the documents, or in very few of them, are unlikely to tell us much about the text or to be of any use in determining the category of a contract. For example, the words "agree" and "agreement" are likely to appear in most of our contracts, and names like "Asriel" are only likely to appear in one or two contracts. Thus, it is common to remove all words that appear in more or fewer than a certain number of documents. Figure 2d shows the bag-of-words representation in figure 2 after removing words occurring very often or seldom. (Note that this will likely also remove stop words if they were not removed explicitly.) This method is also used to control the size of the bag-of-words representation.

**Figure 2d**: the bag-of-words representation in figure 2 after removing words occurring very often or seldom

| | Contract ID 1 | Contract ID 2 | Contract ID 3 |
|---|---|---|---|
| asriel | 0 | 1 | 0 |
| behold | 0 | 1 | 0 |
| bequeath | 0 | 1 | 0 |
| between | 1 | 0 | 1 |
| body | 0 | 1 | 0 |
| booklet | 0 | 0 | 1 |
| born | 2 | 0 | 0 |
| brookhaven | 0 | 0 | 1 |
| … | … | … | … |

It is important to remember that these methods will not all be appropriate in all circumstances. For example, suppose text is encoded for the sake of named entity recognition (extracting the names of the parties involved in a contract, for instance). Removing all words that occur in too few contracts is likely to be rather counterproductive in this situation, unless every counterparty appears in every contract. Experience is key in determining what methods should be used in constructing a given model.

### THE INVERSE DOCUMENT FREQUENCY AND TF-IDF

As we discussed above, one key weakness of the bag-of-words representation is the fact it treats important words laden with meaning (such as "bequeath") on par with more common words devoid of much meaning (such as "the"). We discussed a number of methods for removing the most egregious of these very common words—using a list of *stop words*, and by removing terms that appear too frequently in our set of documents.

This approach, however, only goes so far in resolving the issue. It will remove unimportant words from the bag-of-words representation, but it will do nothing to distinguish the words remaining. For example, the words "bequeath" and "jurisdiction" are both likely to remain after the filters discussed in the last section. The former is very informative, strongly indicating the contract in question is a will, whereas the latter is far more generic in contracts. Nevertheless, both words appearing in a contract would be counted identically in their respective columns in the bag-of-words representation.

To make matters more complicated, different words might have different importance in different sets of documents—consider, for example, the task of looking at a set of wills and identifying those that were drafted by a professional and those that were drafted by an amateur. In this case, every document is likely to contain the word "bequeath." The word

"jurisdiction," on the other hand, is likely to appear more seldom, only in professional documents.

We now discuss the *term frequency-inverse document frequency* (TF-IDF), a method that attempts to solve this problem and—as we will see—can greatly increase the performance of our text mining algorithm. The key insight of TF-IDF is striking in its simplicity; it notes that the more often a word appears in a given set of documents, the less likely it is to have much meaning. Thus, in our example above, "jurisdiction" appears in many contracts and thus isn't very meaningful, whereas "bequeath" only appears in a few contracts.

To formalize this intuition, the TF-IDF approach simply divides every row in the bag-of-words representation by the number of times the word appears in the *entire set of documents*.[iii]

Take the example of the word "and" in the contracts in figure 1a. (We assume that stop words have not yet been removed, to illustrate the TF-IDF approach.) The word appears once in contract 1, and five times in the entire set of contracts. Thus, the entry under "and" in contract 1 would be 1 / 5 = 0.2.

Figure 2e shows the bag-of-words representation in figure 2 after TF-IDF has been applied. Note how words that appear in a document, but appear seldom in the whole set of documents, get a stronger score.

**Figure 2e**: the start of the TF-IDF representation of the contracts in figure 1a

| | **Contract ID 1** | **Contract ID 2** | **Contract ID 3** |
|---|---|---|---|
| a | 0.5 | 0 | 0.5 |
| agreement | 0.33 | 0 | 0.66 |
| agrees | 1 | 0 | 0 |
| all | 0 | 1 | 0 |
| and | 0.2 | 0.4 | 0.4 |
| as | 0 | 0 | 1 |
| at | 1 | 0 | 0 |
| asriel | 0 | 1 | 0 |
| behold | 0 | 1 | 0 |
| bequeath | 0 | 1 | 0 |
| between | 0.5 | 0 | 0.5 |
| body | 0 | 1 | 0 |
| booklet | 0 | 0 | 1 |
| born | 1 | 0 | 0 |
| brookhaven | 0 | 0 | 1 |
| … | … | … | … |

[iii] In reality, there are a number of more complex implementations of TF-IDF that divide the term frequency by a transformation of the document frequency and then normalize the resulting vectors, but the idea remains the same.

## BEYOND THE BAG-OF-WORDS: *N*-GRAMS, FORMATTING, PUNCTUATION, AND ADDITIONAL FEATURES

As we have seen, the bag-of-words and TF-IDF methods capture much of the information in a piece of text, but can omit some key information. In this section, we discuss approaches to including these additional pieces of information in our representation.

First, we consider word order. In the *n-gram* technique, instead of considering single words, we consider groups of $n$ words in building the representation of our document. This requires one row in our representation for every distinct set of $n$ words in *any* contract. Figure 3 shows the 2-gram representation for the phrases "David shall borrow $20 from Ben" and "Ben shall borrow $20 from David." The representations of these two phrases are clearly different—the word order is no longer lost. Of course, one of the major downsides of the technique is the sheer number of rows necessary. If a set of contracts uses, say, 1,000 words, a 2-gram representation could require as many as $1{,}000^2$ rows!

**Figure 3**: 2-gram representation of two sentences

David shall borrow $20 from Ben

| | Frequency |
|---|---|
| 20 from | 1 |
| ben shall | 0 |
| borrow 20 | 1 |
| david shall | 1 |
| from ben | 1 |
| from david | 0 |
| shall borrow | 1 |

Ben shall borrow $20 from David

| | Frequency |
|---|---|
| 20 from | 1 |
| ben shall | 1 |
| borrow 20 | 1 |
| david shall | 0 |
| from ben | 0 |
| from david | 1 |
| shall borrow | 1 |



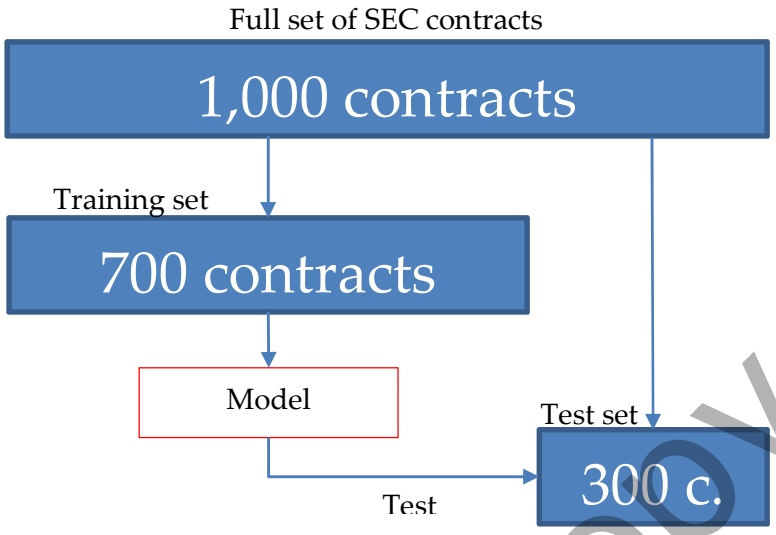
Finally, we mention the possibility of capturing additional information from a document as well as that included in the words therein. This can simply be done by adding rows in the document representations—representing, for example, punctuation, formatting, or the length of the document.

## CLASSIFYING CONTRACTS INTO CATEGORIES—THE PACIFIC TRUST CREDIT UNION

Recall that PTCU's primary challenge was to find compensation and loan contracts in the organization's contract database. SEC data will illustrate a technique that can be used to perform this task.

Specifically, we will split the SEC data into two parts. We will use the first to learn which contracts tend to be loan/compensation contracts, and then test our conclusions on the remaining ones. This is called a *supervised learning technique*. We use existing data to learn our model, and then apply it to unknown data. Figure 4 demonstrates this plan:

**Figure 4**: Training/test plan




Our first step, of course, is to take these SEC contracts and encode them as we described at length above. Specifically, we will:

- Use the bag-of-words representation
- Remove English stop words
- Remove any words that occur in fewer than 10% and more than 90% of documents
- Keep only the 1,000 most frequent words in these contracts

Having encoded the data in this fashion, how can we build a model to determine, based on the encoding, whether the contract is a loan or compensation contract? We will use an algorithm called a *decision tree*—a simple yet powerful classification method that uses a series of "yes/no" questions to determine the category in which the contract falls.

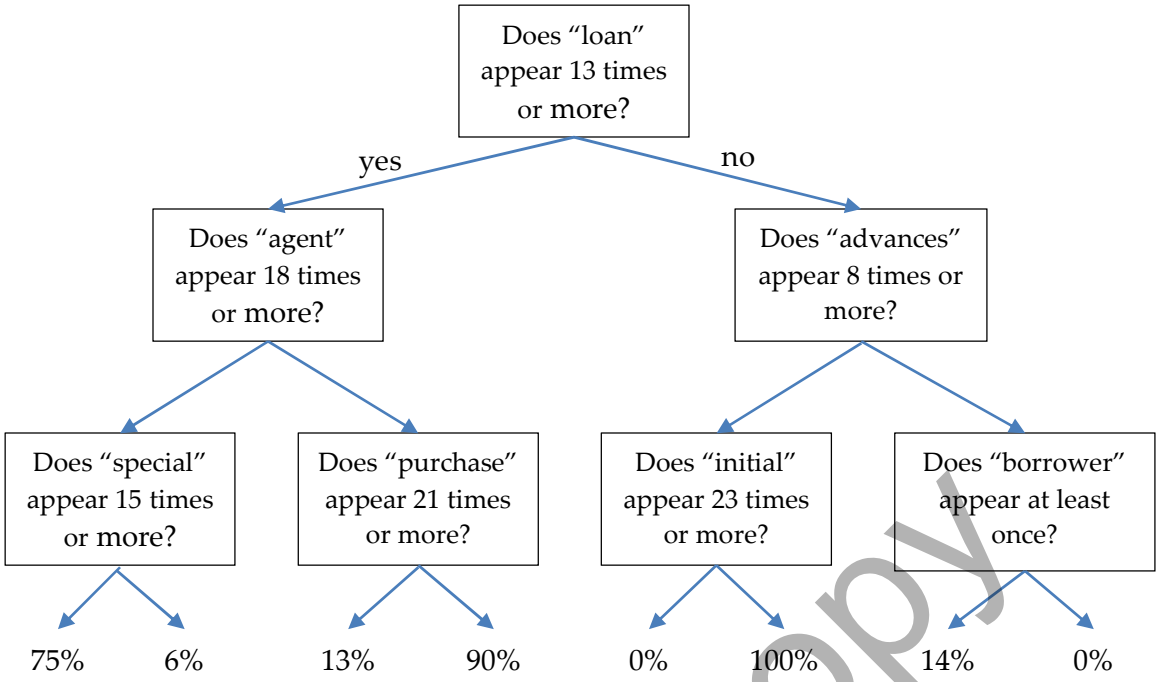
So, for example, we might look at our training data, figure out which word most strongly predicts whether a contract might be a loan, and split the data based on that word, to produce two new data sets. We would then repeat this for each of the two data sets, and so on.[iv]

Applying this method to our training data (the 700 contracts in the diagram above), we can create a tree that predicts the probability that any given contract will be a loan. See figure 5.

**Figure 5**: A decision tree to predict the probability that a contract is a loan



[iv] Of course, continuing this ad infinitum would result in a very large decision tree with many questions. In practice, a number of methods exist to help us decide when to "stop" building the decision tree. These are beyond the scope of this case.




Each level corresponds to a question, and the final numbers correspond to the probability of the contract being a loan. The left branch is always "yes," and the right branch is always "no." For example, according to the tree, a document in which the word "loan" appears 15 times, "agent" appears 10 times, and "purchase" appears 25 times has a 13% chance of being a loan.

How does this tree perform when we try it on the "testing" data (the 300 contracts in the diagrams above)? One way to evaluate this is to use a measure called the AUC (the *area under the curve*). To understand the AUC, consider two contracts—one of which is a loan, and one of which isn't. The AUC is the probability that the contract that *is* a loan gets a higher predicted probability of being a loan than the one that isn't. In short, it's the probability that the model correctly orders the contracts.

For the tree above, the AUC is 88% on the 300 contracts we did *not* use to train the model (the test set).

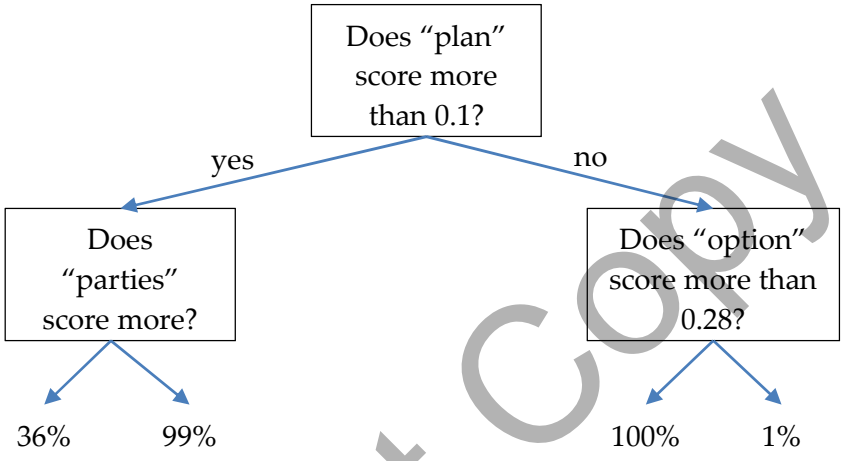
Recall that, as well as identifying loan contracts, the PTCU wanted to identify compensation-related contracts. Applying the same technique to produce a decision tree for these contracts, the model has an AUC of 79% on the test set—not as impressive, but still good.

These models already do a good job at identifying contracts of interest. But how might the PTUC do better? Of course, using a more complex model is one option. But changing the encoding method we use might also help. When we processed the text above, we used the bag-of-words representation without bothering with TF-IDF. Carrying out the TF-IDF transformation, and fitting our trees again, we obtain an AUC of 80% for loans, and 94% for compensation. It is fascinating to see that while TF-IDF is helpful for classifying compensation contracts, it actually *reduces* the performance for loan-related contracts. This highlights the

extent to which encoding text is an art as much as a science, and experimentation is just as important as hard-and-fast rules.

Figure 6 represents the optimal TF-IDF tree for compensation contracts. Note that instead of looking at word frequencies (as in bag-of-words), we look at TF-IDF scores. Notice also how simple the tree is, and yet how well it performs.

**Figure 6**: A decision tree to predict the probability that a contract is a compensation contract, using TF-IDF






**DISCOVERING CATEGORIES IN A SET OF CONTRACTS—THE INVENTOR'S DEN**

Recall that the Den had entered into a wide variety of contracts and sought Evisort's help to classify them into a number of coherent categories.

At first sight, this seems impossible. Unlike PTCU's task, for which there were examples available from which the algorithm could learn, the Inventor's Den is looking for an algorithm to create order from nothing!

Astonishingly, recent advances in artificial intelligence make even this task achievable. The field of *topic modeling* is concerned with precisely this problem—looking at a set of documents and identifying "topics" into which the documents can be sorted.

We will specifically look at a technique called *latent Dirichlet allocation* (LDA). It identifies a set of topics (defined by words most likely to appear inside the topic), and then deduces how much of each topic is present in each document.

For example, on the Inventor's Den data, the model might identify three topics: a "lease" topic (likely to contain words such as "lessor," "property," and "rent"), a "will" topic (likely to contain words such as "testament" and "bequeath"), and an "employment" topic (likely to contain words such as "employee," "senior," and "junior"). Figure 7a represents these topics. (Note that, at this stage, we have not yet discussed *how* LDA might determine these topics.)

**Figure 7a**: a representation of the topic matrix for a potential set of topics present in a document

| | Topic 1 | Topic 2 | Topic 3 |
|---|---|---|---|
| lessor | 0.1 | 0.000 | 0.000 |
| property | 0.1 | 0.001 | 0.000 |
| rent | 0.2 | 0.000 | 0.000 |
| testament | 0.01 | 0.2 | 0.001 |
| bequeath | 0.023 | 0.15 | 0.0001 |
| employee | 0.003 | 0.001 | 0.18 |
| junior | 0.01 | 0.0001 | 0.1 |
| senior | 0.04 | 0.1 | 0.1 |
| … | … | … | … |

Note that each entry in figure 7a corresponds to the probability that a word linked to the topic in the column will be the word in the row (for example, a word linked to topic 1 has a 20% chance of being "rent"). Each column must sum to 1, as all probabilities do (in this case, the columns sum to less than 1 because other words have been omitted from the table). Note that each topic is not automatically given a "name"; the meaning of each topic must be inferred from the names therein. Note also that while each topic has a set of "high-probability words," there is a nonzero probability other words might appear in that topic as well.

Once LDA has identified the topics present in a document, it then automatically assigns each document to one of these topics (again, we have yet to describe how). Figure 7b shows an example of such an assignment for the documents listed in figure 1a at the start of this case.

**Figure 7b**: a representation of topic assignments made by LDA

| | Topic 1 | Topic 2 | Topic 3 |
|---|---|---|---|
| Document 1 | 0.8 | 0.15 | 0.05 |
| Document 2 | 0.02 | 0.98 | 0.000 |
| Document 3 | 0.25 | 0.000 | 0.75 |
| … | … | … | … |

Note that document 1 in figure 7b , for example, is 80% topic 1, 15% topic 2, and 5% topic 3. Note that the numbers in each row sum to 1. This flexible representation allows LDA to reflect the reality that documents rarely fall into one category perfectly; a will, for example, might bequeath a property currently occupied by tenants. Similarly, an employment agreement might include providing the employee with housing.

Once we have these two tables of numbers—words in each topic (figure 7a) and topics in each document (figure 7b)—our task is complete. We have a full description of the topics present in our set of documents, and an idea of which document belongs to which topic.

Of course, it remains to discuss how these tables of numbers are actually obtained. The next section explains this algorithm, but it is somewhat technical; it can be skipped without any loss of continuity.

### THE MATHEMATICS BEHIND LDA

The details of LDA are highly mathematical, but the idea behind it is striking in its ingenuity and simplicity. Before we discuss it in greater detail, it is worthwhile to consider a simpler example that will shed light on the methodology involved.

Our simpler example involves a process in which two team leaders need to be chosen for a class project. A hundred students are randomly selected on campus and locked in a room. Some are humanities students, and some are science students—overall, the room ends up containing a proportion $p$ of humanities students. These students are then asked to pick two team leaders, who are duly selected. You then come back, and observe that one of the leaders chosen was a humanities student, whereas the other was a science student.

Based on these observations, we want to find $p$, the true proportion of the room that comprised humanities students. Now of course, we might already have a rough idea of what $p$ is, based on the composition of the campus. (Is it a liberal arts college? A technological university?) This is our *prior* knowledge of the number $p$. But the aim here is to update this *prior* belief using our *observed* information (the two team leaders chosen).

What is the analogy to our text mining application? The outcome (one science team leader, one humanities team leader) is the observed data, and it corresponds to the text of the documents we observe (for example, the documents in figure 1). The underlying parameter (the proportion $p$ of humanities students in the room) is the parameter we want to find, and it corresponds to the topic tables in figures 7a and 7b. Finally, the *prior* information (our general knowledge of $p$ based on the kind of college campus) is our general knowledge of how the English language works, which will necessarily inform the composition of the tables in figures 7a and 7b. Just as we want to use the chosen leaders to inform our knowledge of $p$, we want to use the contract texts to inform our knowledge of the topic tables in figures 7a and 7b.

How would we go about doing this? One way is to use a so-called *generative model*. The basic idea is to hypothesize how the underlying parameters led to the outcomes, and then use that hypothesis to figure out how the observed outcomes will affect our knowledge of the underlying parameters.

In the class example, our generative model should describe how the underlying parameter $p$ arises (the prior information), and then how the observations (one humanities leader, one science leader) arise as a result of that probability. The model might look something like this:

a) Pick out students randomly from campus, to get our room of students with a proportion $p$ of humanities. The composition of the campus would provide *prior* information about $p$.
b) Select two student leaders from the room. We will assume this happens *randomly*.

c) Observe the result.

It's crucial to realize that for a generative model to work, our assumptions as to how the data were generated do not have to be exactly correct. The more correct they are, the better our generative model will perform, but often a tentative model is good enough. In the example above, it seems unlikely that the two team leaders would have been chosen at random from the students in the room. One could hypothesize that humanities students might have had more of an edge because of their oratory skills. We could improve the model by taking that into account, but it's not essential.

In our text example, both data and underlying characteristics are more complicated, so the generative model is also quite complicated. LDA uses the following generative model:

a) Randomly[v] pick numbers for the tables in 7a and 7b. The exact way these random numbers are chosen (see footnote) encodes our *prior* information.
b) Generate the documents as follows:
   - For every word in every document, randomly pick a topic based on the probabilities in 7b…
   - …then generate a word from that topic using the probabilities in 7a.
c) Observe the resulting documents.

It is, once again, important to realize that this generative model is by no means exactly correct. If writing a contract matching a certain topic were as simple as selecting random words from random topics, as in step (b), lawyers would quickly be out of business! And many more complex generative models beyond LDA have been used and are used for text modeling. Nevertheless, despite the seemingly ludicrous assumptions LDA makes, we will see it works very well.

Once we have a generative model, we are finally able to use our observed outcomes (one humanities and one science team leader) to refine our knowledge of the underlying parameter $p$.

In particular, suppose I wanted to ask, "What is the probability $p$ is ¼ given my observed data?" We could calculate this number as follows:[vi]

$$\begin{aligned}&\mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{4} \text{ given I observed one science and one humanities}\right)\\ &\quad\approx \mathbb{P}\text{rob}\left(\text{I observe one science and one humanities given } p \text{ was } \tfrac{1}{4}\right)\\ &\qquad\times \mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{4} \text{ given the kind of college campus I'm on}\right)\end{aligned}$$

[v] These initial random numbers are derived from a mathematical distribution called the Dirichlet distribution, hence the name latent *Dirichlet* allocation. Just as the kind of college campus encodes our prior belief about $p$ (the number of humanities students in the room) the Dirichlet distribution encodes our prior belief about topics and documents *in general*, before we even look at that example. Part of LDA's success stems from how flexible the Dirichlet distribution is. It allows us to reflect, a priori, whether we expect topics to include many words or few words, for example, and how many topics we expect each document to include.

[vi] Those familiar with Bayes' theorem will recognize this result as a simplified version of it.

The first probability is trivial to calculate:[vii]

$$\begin{aligned}
&\mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{4} \text{ given I observed one science and one humanities}\right) \\
&\quad = \mathbb{P}\text{rob}(\text{Humanities given } p \text{ was } \tfrac{1}{4}) \\
&\qquad\quad \times \mathbb{P}\text{rob}(\text{Science given } p \text{ was } \tfrac{1}{4}) \\
&\quad = \tfrac{1}{4} \times \left(1 - \tfrac{1}{4}\right) \\
&\quad = \tfrac{1}{16} \\
&\quad = 0.1875
\end{aligned}$$

The second probability reflects our *prior* knowledge of $p$, the proportion of humanities students in the room. If we were at MIT, we might expect the probability $p$ = ¼ to be reasonably low, since there are so few humanities students on campus. At a more evenly balanced college, we might expect that probability to be higher. For the sake of this calculation, we will assume we know nothing about the college campus in question, and so we will simply omit this last term, and get

$$\mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{4} \text{ given I observed one science and one humanities}\right) = 0.1875$$

We can carry out the same steps with other values of $p$, and find, for example, that

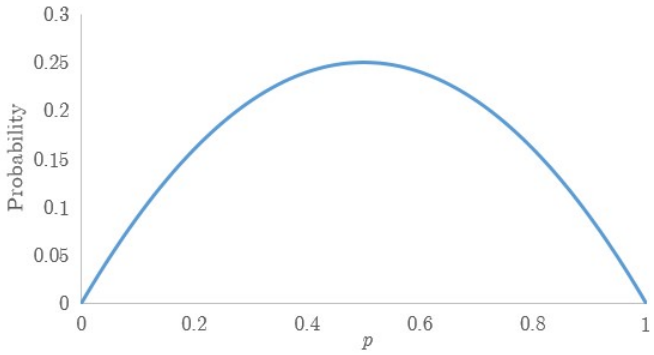
$$\mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{3} \text{ given I observed one science and one humanities}\right) = 0.22$$
$$\mathbb{P}\text{rob}\left(p \text{ is } \tfrac{1}{2} \text{ given I observed one science and one humanities}\right) = 0.25$$

Doing this for every possible value of $p$ and plotting the resulting probabilities, we get the graph in figure 8. Unsurprisingly, the *highest* probability is at $p$ = ½. This is reassuring. Since we observed one science and one humanities student, it makes sense that the most likely value of $p$ is ½.

**Figure 8**: For each possible value of $p$, this graph shows the probability that $p$ was indeed the true proportion of humanities students given the observed data.



[vii] In reality, once the first student is picked—say, a humanities student—there is one less humanities student in the room, and the calculation should be adjusted to take this into account. But in a room of 100 students, the difference would be minimal.




We can use this graph to finally answer our question: "Based on our observed data, what is *p*?" There are a number of ways we could use the graph in figure 8 to answer that question—the easiest is just to pick the highest-probability value,[viii] which is ½.

How does this translate to the text application? How can the LDA generative model be used to find the probability of a set of tables, 7a and 7b, given the contract texts? Unfortunately, and perhaps unsurprisingly, the calculations are considerably more complicated. See Appendix for a small part of these calculations.

We now turn back to the example of the documents in figure 1 and the set of topics and topic assignments in figures 7a and 7b. The algorithm (part of which is listed in the Appendix) will try to find the probability that tables 7a and 7b were correct, given the documents in figure 1. This might proceed as follows:

- Document 1 seems to mostly be assigned to topic 1 (first row of figure 7b), which puts a high probability on the words "lessor," "property," and "rent" (first column of figure 7a). The document itself does contain these words, so the probability of these parameters in 7a and 7b is high.
- Document 2 seems to mostly be assigned to topic 2 (second row of figure 7b), which puts a high probability on the words "testament" and "bequeath" (second column of figure 7b). The document itself contains these words, so the probability of these parameters in 7a and 7b is high. However, topic 2 also places relatively high weight on the word "senior," which does not appear in document 2, thus lowering the probability.
- Etc.

Once the probability of each table is calculated, we can again find the most likely.



[viii] In reality, the algorithms would not return the "most likely" value but would instead use the full distributions. The details are less important for our purposes.

It remains to ask what happens when we apply this technique to the corpus of documents from the SEC. We choose to run an algorithm with four topics,[ix] and use the method described above to find the topic distributions and document allocations in figures 7a and 7b. We then find the highest probability words in each of the resulting topics. The results are shown in figure 9.

**Figure 9**: the most important words in each topic, as determined by latent Dirichlet allocation

| Topic 1 | Topic 2 | Topic 3 | Topic 4 |
|---|---|---|---|
| stock | tenant | borrower | Company |
| shares | landlord | lender | Section |
| company | lease | agent | Seller |
| common | premises | section | Party |
| corporation | lessee | loan | closing |
| securities | lessor | credit | executive |
| warrant | rent | administrative | purchaser |
| exercise | property | time | business |

At first glance, the method is very successful. It seems to have identified four distinct topics in the corpus:

- Topic 1 appears to relate to the granting of stock options.
- Topic 2 appears to relate to leases.
- Topic 3 appears to relate to borrowing and/or lending.
- Topic 4 appears to relate to sales of some sort.

The method also assigns each contract to a combination of these categories.

Applying this technique to the Inventor's Den's contracts, perhaps with more topics, would result in a number of categories that could be used to further investigate the contracts.

## Conclusion

We examined the ways text data can be encoded for use by artificial intelligence algorithms. We then considered the way these artificial intelligence algorithms could be used to automatically classify contracts into their categories, and even to find structure within the contracts themselves (for example, to identify the counterparties in the contract, or the date on which the contract was executed).

[ix] In this case, we arbitrarily choose to run an algorithm with four topics, but approaches exist to decide exactly how many topics should be used.

Needless to say, we have only scratched the surface of the vast universe of things that can be done with text data using artificial intelligence. To start, there are many more methods for text preparation beyond those mentioned in this case. But most important, there is a wide range of other methods available to analyze the text once it is encoded.

Relevant to Evisort in particular is the field of *named entity recognition*, which attempts to learn the *function* of each word in a piece of text—for example, named entity recognition might be able to automatically identify the counterparties in a contract, or the contract's expiration date.

Beyond named entity recognition, methods such as word2vec extract logical meaning from words, and will group together words that share the same context. For example, word2vec might represent the word "jurisdiction" and the various jurisdictions mentioned in contracts together, because they tend to appear in the same context in a contract.

Finally, once these methods are used to extract meaning from the text, a number of methods exist to further analyze the results. For example, comparing the topics present in a company's contracts in 2005 to those in 2015 could provide valuable insight into the way the company's operations have changed over that period.

One thing is certain: The amount of text data available in a wide variety of industries is only growing, and it will become increasingly crucial for companies to grasp the methods required to extract value from these data. The methods discussed in this case provide a good starting point in gaining an understanding of what is possible, and how it can be achieved.

## Appendix

A small part of the calculations involved in finding the probability of a set of parameters, given observed documents.

$$\mathbb{P}\left(\boldsymbol{Z},\boldsymbol{W};\alpha,\beta\right)=\int_{\varphi}\prod_{i=1}^{K}\mathbb{P}\left(\varphi_{i};\beta\right)\prod_{j=1}^{M}\prod_{t=1}^{N}\mathbb{P}\left(W_{j,t}\mid\varphi_{Z_{j,t}}\right)\,\mathrm{d}\varphi\int_{\theta}\prod_{j=1}^{M}\mathbb{P}(\theta_{j};\alpha)\prod_{t=1}^{N}\mathbb{P}\left(Z_{j,t}\mid\theta_{j}\right)\,\mathrm{d}\theta$$

$$\begin{aligned}
&\mathbb{P}\left(Z_{(m,n)}=k\mid\boldsymbol{Z}_{-(m,n)},\boldsymbol{W};\alpha,\beta\right)\\
&\quad\propto\mathbb{P}\left(Z_{(m,n)}=k,\boldsymbol{Z}_{-(m,n)},\boldsymbol{W};\alpha,\beta\right)\\
&\quad=\left(\frac{\Gamma\left(\sum_{i=1}^{K}\alpha_{i}\right)}{\prod_{i=k}^{K}\Gamma(\alpha_{i})}\right)^{M}\prod_{j\neq m}\frac{\prod_{i=1}^{K}\Gamma(n_{j,(\cdot)}^{i}+\alpha_{i})}{\Gamma\left(\sum_{i=1}^{K}n_{j,(\cdot)}^{i}+\alpha_{i}\right)}\left(\frac{\Gamma\left(\sum_{r=1}^{V}\beta_{r}\right)}{\prod_{r=1}^{V}\Gamma(\beta_{r})}\right)^{K}\\
&\qquad\prod_{i=1}^{K}\prod_{r\neq v}\Gamma\left(n_{(\cdot),r}^{i}+\beta_{r}\right)\frac{\prod_{i=1}^{K}\Gamma\left(n_{m,(\cdot)}^{i}+\alpha_{i}\right)}{\Gamma\left(\sum_{i=1}^{K}n_{m,(\cdot)}^{i}+\alpha_{i}\right)}\prod_{i=1}^{K}\frac{\Gamma\left(n_{(\cdot),v}^{i}+\beta_{v}\right)}{\Gamma\left(\sum_{r=1}^{V}n_{(\cdot),r}^{i}+\beta_{r}\right)}\\
&\quad\propto\prod_{i}\Gamma\left(n_{m,(\cdot)}^{i,-(m,n)}+\alpha_{i}\right)\prod_{i}\frac{\Gamma\left(n_{(\cdot),v}^{i,-(m,n)}+\beta_{v}\right)}{\Gamma\left(\sum_{r=1}^{V}n_{(\cdot),r}^{i,-(m,n)}+\beta_{r}\right)}\left(n_{m,(\cdot)}^{k,-(m,n)}+\alpha_{k}\right)\frac{n_{(\cdot),v}^{k,-(m,n)}+\beta_{v}}{\sum_{r=1}^{V}n_{(\cdot),r}^{k,-(m,n)}+\beta_{r}}\\
&\quad\propto\left(n_{m,(\cdot)}^{k,-(m,n)}+\alpha_{k}\right)\frac{n_{(\cdot),v}^{k,-(m,n)}+\beta_{v}}{\sum_{r=1}^{V}n_{(\cdot),r}^{k,-(m,n)}+\beta_{r}}
\end{aligned}$$

*Note*: The details of these calculations fall outside the scope of this case.