Making Sense:  
Extracting Meaning from Text by Matching Entities and Terms to Ontologies and Concepts  

Chapter 6  
Feature Vectors for Term-to-Concept Mapping  

Alianna J. Maren  
Northwestern University School of Professional Studies  

Chapter Draft: 2015-07-30
6.1 Why Matching Terms to Concepts Is Important

One of the most important steps in preparing a text analytics system is to carefully define both ontologies and concept classes. Members of the text analytics community understand the value of each process, and various open source and Commercial-Off-the-Shelf (COTS) systems support either or both ontology and concept class definition.

What is less well understood - although potentially providing great value - is defining a mathematically-based means for specifying and differentiating concept classes associated with neighboring ontological items. When we do this in a satisfying way, we can then create a text analytics system that will identify precisely and cleanly (with minimal overlap) the concept classes (and thus the corresponding ontological items) associated with a given Data Source Item (DSI), such as a given article or blogpost.

The implications of getting clean (well-separated) identification of concepts and ontological items associated with a given DSI are threefold:

- **Precise associations** - identification of which other concepts and ontological items are associated with the ones used to identify a sub-corpus,

- **Precise sentiment** - association of sentiment with specific concepts/ontological items, and

- **Novelty identification** - being able to determine when new concepts and/or ontological items occur associated with an existing concept.

6.2 High-Level Approach Overview

The following Figure 6.1 illustrates our overall approach.

Previous chapters have given us a methodology for extracting entities and noun phrases from raw text, and for coalescing these into a set of terms. Certain terms may represent equivalence classes; sets of synonyms and/or phrases that essentially mean the same thing. Practically speaking, we can often reduce the number of raw extracted entities and noun phrases by about 50% by carefully constructing our equivalence classes. All of this still left us with a large set (or a Feature Vector) of many terms. These steps are shown within the pale blue box in the upper left-hand corner of Figure 6.1.
Creating Well-Separated Ontological Concepts

Figure 6.1: Creating well-separated ontological concepts.
However, this collection of terms was still a signal-level (Level 1) data representation. It didn’t give us any abstraction; no matter how well-consolidated, collections of terms don’t give us insight into how a DSI’s content relates to our world view.

While we were doing statistical-level analyses, we were also developing our ontology, or world-view. This was our symbol-level (Level 5) knowledge representation. Our ontology expressed how we thought the content of our DSIs was organized.

A key step here is that we are going to define the notion of concepts. Right now, we’re representing each ontology node as a single concept; a one-to-one mapping. We can get more complex later, but this gives us a simple starting point.

These steps defining ontologies and mapping concepts to ontologies are shown within the pale purple box in the upper right-hand corner of Figure 6.1.

At the conclusion of the last chapter, we still didn’t have a good way to connect our Term Feature Vector to our ontology.

Our goal in this chapter is to define a reliable, repeatable means for mapping subsets of terms to the corresponding ontology nodes. This is the process shown in the mid-level mottled grey box in the center of Figure 6.1.

One thing that we’re noting now (but will defer to much later in this book) is that once we’ve created an initial Feature Vector-to-Concept mapping, we can progressively refine this map. Refinements will give us more precise concept differentiation. It will allow us to state, with both greater accuracy and greater precision, which concepts are in which DSIs. These steps are represented at the bottom of Figure 6.1; we will discuss how they can be accomplished later in this text.

The result of this whole process is that we will have transformed the way in which represent our DSIs. Instead of having a bag of words, or even a well-ordered set of terms with their associated term frequencies, we will now be able to represent each DSI in terms of which concepts are present, to which strengths.

We can then use some of the same mathematical techniques that we would have used on the Term Feature Vector representing each DSI. Instead, though, we’ll have a Concept Feature Vector. Instead of term strengths, we’ll have concept strengths. This gives us a much more compact means of representing each DSI’s content, and a more powerful means of assessing just what is in our data corpus.
6.3 Concept Definition: Flat-World versus Structured-World (Ontology)

6.3.1 A Point of Reference: The IBM SPSS Text Analytics Approach to Concept Formation

Most advanced text analytics systems have some form of concept extraction in addition to basic entity extraction. One example is IBM’s CRISP-DM system. Their 32-page basic documentation overviews processing steps, including extracting concepts, determining equivalence classes, and co-occurrence [1].

The following passages, taken from the IBM SPSS Text Analytics manual for CRISP-DM, outline IBM’s approach to both concept extraction and ontology creation. These steps are given as a form of baseline; note that the opportunity to mathematically refine concepts is limited.

Certain commercial systems offer analysts the ability to build concepts (categories). As a point of comparison, the final major step in the CRISP-DM system is that analysts can build categories. An extract from the CRISP-DM manual specifies their process.

**IBM SPSS Processing Steps: Building Categories**

In all cases, however, the classification process is iterative: a researcher applies certain techniques, evaluates the results, makes changes either to the technique chosen or to the resultant categories and refines the results... Both automated and manual classification techniques are available with IBM SPSS solutions. The automated, linguistics-based techniques available include:

- Concept derivation,
- Concept inclusion,
- Semantic networks, and
- Co-occurrence rules.
6.3.2 How Our Approach to Concept-Building Differs from Most Commercial Systems

It is clear that with CRISP-DM (TM), building categories is a complex process, and allows the developers and knowledge managers substantial input.

The problem with this approach, though, is that all concepts are at the same hierarchical level. It is a sort of flat-world view of concepts, and correspondingly, of ontologies.

What we want is a more hierarchical, structured world.

We want to be able to specify that some concepts (and their corresponding parent ontology nodes) are more abstract and general than their children nodes. We also want to be able to mathematically measure the degree to which children nodes of the same parent are distinguished from each other, and to what extent they overlap with their parent and grandparent nodes.

Although we can measure concept-to-concept distances in a flat-world representation, we can’t specify the kinds of distances that we want in terms of a structured world-view.

This is why we’re taking the careful approach outlined in this chapter. It ultimately gives us greatest control over mapping DSIs to our world-view.

6.4 Notation and Naming Conventions

6.4.1 Establishing Notation

Level 1: Extracted and Entities, Equivalence Classes, and Terms

Suppose that we have a reference (training and ground-truthed) training corpus of $M$ Data Source Items (DSIs), and that we extract a set of $N$ nouns, noun phrases, and other phrases.

We perform a set of processing steps to generate, from this list of extracted nouns and noun phrases. On this set of extractions, we perform a number of other steps, including formation of equivalence classes. At the end, we have a total of $J$ distinct terms extracted from this corpus. We use these terms to create a Term Feature Vector of length $J$. Each spot in the feature vector is reserved for a different term. The value that is stored in each location represents the strength of the term, and is usually the frequency for that term’s appearance.

We we’ve defined our collection of terms, we can create a Term Feature
Vector for a collection of any size of DSI’s; we can have a Term Feature Vector for any single specific DSI, and we can have a Term Feature Vector for an entire corpus.

Some of these terms will define entities; specific objects or other things that we can identify. Some will define more abstract objects. For our purposes, a term may be a single word or a phrase, and we will assume that terms include equivalence classes, so that synonyms and other means of expressing the same thing are mapped into the same representative term.

Level 5: Concepts and Ontologies

Suppose further that we build an ontology composed of \( K \) ontological nodes, and have also defined as set of \( I \) concepts. For now, we will have a one-to-one association between concepts and ontology nodes, so that \( I=K \); this will not necessarily be the case in more advanced work.

The notion of a concept here is broad; it includes entities as well as abstract concepts.

<table>
<thead>
<tr>
<th>Item</th>
<th>Set Notation</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source Items (DSIs)</td>
<td>( M )</td>
<td>( m=1 \ldots M )</td>
</tr>
<tr>
<td>Extracted Nouns and Noun Phrases</td>
<td>( N )</td>
<td>( n=1 \ldots N )</td>
</tr>
<tr>
<td>Terms</td>
<td>( J )</td>
<td>( j=1 \ldots J )</td>
</tr>
<tr>
<td>Concepts</td>
<td>( I )</td>
<td>( i=1 \ldots I )</td>
</tr>
<tr>
<td>Ontology Nodes</td>
<td>( K )</td>
<td>( k=1 \ldots K )</td>
</tr>
</tbody>
</table>

We also define these same expressions in terms of sets; we say that we have the set \( M \) of DSIs, the set \( N \) of nouns and noun phrases, the set \( J \) of feature vector elements composed of terms, the set \( I \) of concepts, and the set \( K \) of ontological elements.

6.4.2 Naming Conventions

In order to have a naming convention that allows us to easily parse ontology nodes and concepts by their names, we use underscores to separate the words in the names. Our convention will be to use lower-case letters exclusively as well as italics for both ontology nodes and their corresponding concepts.

For example, from our terms collection, Hillary Clinton would normally be an entity. Hillary Clinton also becomes an equivalence class. The
extracted entity *Clinton* would be matched against the *Hillary Clinton* entity in our set of equivalence classes, if the material surrounding the *Clinton* extraction provide sufficient matching evidence.

For our purposes, we are making *hillary_clinton* into a concept, because we want to apply the same methodology throughout.

Other terms that could identify the concept *Hillary Clinton* are *Secretary of State* and *putative front-runner for the Democratic candidate for the 2016 presidential election*. We would also include those terms that map onto these terms as syntactic variants.

### 6.5 Process Overview

**Preliminary Steps - Already Accomplished**

- **Form feature vector** - define a Feature Vector of all the terms occurring in the corpus; this Feature Vector (FV) will have $J$ items in it,
- **Identify terms** - for each concept $i$ in a set of $I$ concepts, select those terms that are most associated with concept $i$,

**Steps for each concept $i$ of a set of $I$ concepts**

- **Select DSIs for concept definition (first pass)** - for each concept $k$, identify a subset of DSIs that seem to include concept $i$, there will likely be overlap - a given DSI will typically include several concepts and can be used to define each of those concepts,
- **Select terms for concept definition (first pass)** - from the subset of DSIs selected for concept $i$, identify all the terms that seem to reference concept $i$ - this may be a much more extensive list than simple synonyms; again there may be overlap, a given term can be used to indicate the presence of more than one concept,
- **Identify term frequency per DSI (first pass)** - for each DSI from the subset of DSIs selected for concept $i$, identify the term frequency of those terms that seem to define or be related to concept $i$,
- **Identify term frequency distribution for each term** - for each term $j$ selected for concept $i$, identify the *distribution of term frequencies* as well as the expected (median) frequency; the expected frequency
will be used to define the Reference Feature Vector (RFV) for concept $i$; it is $RFV(j, i)$.

Figure 6.2: Matching a DSI against a concept.

Determine extent to which concept $i$ is present in a given DSI

- **Find Feature Vector Difference** - for a given DSI, obtain its associated Term Feature Vector (TFV), and compute a difference measure between the DSI’s TFV and the CFV for concept $i$; $RFV_i$; the cosine method for vector differencing is a common one, and then

- **Apply scaling/truncation** - it may be useful to either apply a scaling factor to the resulting difference, or to have a cut-off threshold for identifying whether concept $k$ is present in that DSI.
6.6 Dimensionality Reduction

6.6.1 Relative Set Sizes

Our goal is to represent the content in a data corpus at increasing levels of abstraction. The reason for this is that at the lowest level of abstraction, we have a large number of specific different nouns and noun phrases. They are often similar in meaning, and they are not independent from each other. This means, when we see one noun or noun phrase, we are also likely to see related ones.

We could try to represent the corpus content using a \( N \times M \) matrix, where each column of length \( N \) represents an initial feature vector how often a given noun or noun phrase occurs in one of the \( M \) DSIs. (Each noun or noun phrase here would be a specific feature vector element; corresponding to a row within the feature vector.)

This is a very unwieldy way of representing corpus content.

Typical Set Sizes: Noun Phrases and Terms

Typically, the set of nouns and noun phrases (\( N \)) has the largest number of elements in it, compared to the other sets. A relatively small corpus of order \( 10^x \) can easily generate about \( 10^{x+2} \) nouns and noun phrases. (This is assuming that the corpus is composed of DSIs with about 500+ initial words, before any preprocessing takes place.)

If the corpus has many similar or identical concepts, then increases in corpus size will produce a slower growth rate in the number of distinct new nouns and noun phrases. Typically, though, we still have \( N >> M \).

Our first step in simplification is to create equivalence classes, where we identify those nouns and noun phrases that mean essentially the same thing.

We call the resulting set of nouns and noun phrases (including those that represent equivalence classes) terms.

With careful work, we can usually reduce a set of \( N \) nouns and noun phrases to about \( J = N/2 \) terms; a 50% reduction in set size. This is good, but we are still left with a very large number of terms.

Typical Set Sizes: Concepts and Ontologies

We use ontologies to express our world view. Because our world view is large and complex, it is not unreasonable that ontologies will consist of very
many nodes. These nodes should be well-organized with regard to each other. Nodes under the same parent should have the same kind of relationship to the parent. They should share some attributes of their parent (and other ancestral nodes, going up an ontology-tree), but they should also be more specific than the parent.

Nodes under the same parent should also be distinct from each other. It is not unrealistic that even a very small corpus (order $10^2$) could have an associated ontology of substantially more nodes ($10^2 - 10^{2+2}$), where the ratio of ontology nodes to corpus items becomes much smaller as the corpus size increases, especially if the corpus is tightly controlled to be about a specific subject area (modeled with a well-defined ontology).

6.7 Formal Method for Matching Terms to Concepts

For now, we model ontology nodes by identifying a unique concept for each node (a 1:1 mapping of concept:node). Later, we can change this approach. For now, it is the simplest starting place.

We model the concepts by creating associations of terms to concepts. That is, we determine the degree to which a certain DSI discusses a given concept (ontology node) by finding terms that are associated with that concept.

Preferably, the ontology nodes (concepts) are distinct from each other. In other words, if two nodes are children of the same parent node, the terms that define each node are different enough so that we can map document content to specific ontology nodes (actually, to their corresponding concepts) with little ambiguity. (One document can, of course, map to multiple ontology nodes. This is expected. It can also map to children of the same parent, as well as the parent itself.)

Our goal is to find a mathematical means of generating the mappings from a set of terms to a set of ontology nodes. Ideally, we also find a means of automating the mapping definition.

6.7.1 C: The Term-to-Concept Mapping Matrix

We have $I$ concepts, and can represent this with a Concept Vector $P$ of length $I$, where the $i^{th}$ element of the vector corresponds to Concept $i$, or $P_i$. 
Similarly, we have \( J \) terms, and can represent these with a Term Vector \( T \) of length \( J \), where the \( j^{th} \) element of the vector corresponds to Term \( j \), or \( T_j \).

To map from the Term Vector \( T \) (length \( J \)) to the Concept Vector \( P \) (length \( I \)), we need \( C \), an \( i \times j \) mapping matrix.

\[
\begin{bmatrix}
P_1 \\
P_2 \\
\vdots \\
P_i \\
P_J
\end{bmatrix} =
\begin{bmatrix}
c_{1,1} & c_{1,2} & \cdots & c_{1,J} \\
c_{2,1} & c_{2,2} & \cdots & c_{2,J} \\
\vdots & \vdots & \ddots & \vdots \\
c_{i,1} & c_{i,2} & \cdots & c_{i,J} \\
\vdots & \vdots & \ddots & \vdots \\
c_{I,1} & c_{I,2} & \cdots & c_{I,J}
\end{bmatrix}
\begin{bmatrix}
T_1 \\
T_2 \\
\vdots \\
T_j \\
\vdots \\
T_J
\end{bmatrix}
\] (6.1)

So far, we have not defined exactly what the elements of either the Concept Vector \( P \) or the Term Vector \( T \) really mean; that is, we have not defined them to be either for the full corpus or for a particular document (DSI).

The truth is that we can define both of these vectors two different ways. In each case, and for each vector element (whether \( P_i \) or \( T_j \)), we want to think of each element as representing a strength to which that concept or term is represented in something. It could be the strength in a particular DSI, or it could be its strength in the overall corpus, or even a specific subcorpus.

If we want to represent term strength, we could use the term frequency. We don’t want to use the \( tf-idf \) (term frequency times inverse-document-frequency) here; if we’re going to multiply the term frequency by the inverse term frequency, we can do so by putting the inverse term frequency into the appropriate coefficients \( c_{i,j} \).

If we want to represent concept strength, we could think of how much that concept shows up in a given DSI, or how much it shows up in a corpus. The mathematics will be very similar.

Thus, we begin by thinking about concept strength for a particular concept \( P_i \).

\section*{6.7.2 \( R \): The Reference Feature Vector for a Given Concept}

Suppose that instead of thinking about the full set of \( I \) concepts, we want to focus our attention on a particular concept \( P_i \). To obtain the strength for \( P_i \) in \( \mathcal{I} \) \((i \in \mathcal{I})\), we have multiplied the \( i^{th} \) row in matrix \( C \) by the Term Vector
This multiplication, of a $1 \times J$ vector (row vector $i$ from matrix $C$) by a $J \times 1$ vector $P$ gives us a $1 \times 1$ result, or a scalar. This scalar number is the strength of concept $P_i$. If the Term Vector $T$ gives the various term strengths in a single DSI, then we obtain the overall concept strength for $P_i$ in that DSI. If the Term Vector $T$ gives the various term strengths for a set of DSIs (up to and including a whole corpus), then we have obtained the strength with which concept $P_i$ shows up in that set.

While this approach can be made more complex, dealing with interactions among various documents, we can safely (at least for now) concentrate on developing a method that gives us the strength of concept $P_i$ when we have a Term Vector $T$.

**More specifically, we can start to think about what a reference vector for Concept $P_i$ would look like.**

What we are asking here is the reverse question to what we were asking in the previous subsection. Earlier, we were asking ourselves, “How strong is concept $P_i$ in this (set of) DSI(s)?”

Now, we are asking, “If this concept were represented at maximal strength in this (set of) DSI(s), how would it show up as the appearance of different terms?”

This means that for each concept $P_i$, we want to build a Reference Feature Vector (RFV); $R_i$, which is a vector of length $J$, because we have $J$ terms. To specify the term $j$ of $R_i$, we use $R_{i,j}$.

### 6.7.3 The Reference Vector $R$ Is a Sparse Vector

The Reference Feature Vector $R_i$ for a given concept $P_i$ is a sparse vector. That means that it has many zeros in it.

If we think about it, this is very reasonable.

The Reference Feature Vector $R_i$ is length $J$. However, relatively few terms actually relate to any given concept.

For even a very small data corpus, there can be hundreds or even thousands of terms. There can also be hundreds or even thousands of concepts. (And this is true even for a very small corpus!)

Of the hundreds, thousands, or potentially tens and hundreds of thousands of terms, only a few are relevant to a particular concept. This means that the relevance strength for all the non-relevant terms should be zero. We simply don’t care how much other terms show up; they’re not relevant to our given concept, so they’re not a part of our computation.
6.7.4 The Two Questions on Concept Strength are Related

We have been asking ourselves two different questions:

1. What is the strength of this particular concept \( P_i \) in a particular (set of) DSI(s), \textbf{given} that we have the strengths of different terms showing up in this (set of) DSI(s), and

2. Given that a particular concept \( P_i \) is showing up at full strength in this particular (set of) DSI(s), what would the various term strengths be?

Interestingly enough, if we can answer the second question, we can also answer the first.

That is, if we know what the term strengths are like when a given concept is present at full strength, then we can tell to what degree that concept is present in a given DSI, or set of DSIs.

We would express these term strengths for concept \( P_i \) in a vector \( R_i \), which would be \( J \) units in length, because there are \( J \) total terms.

For the rest of this chapter, we will present the approach as though we were modeling concept strength in just one DSI, because the extension to a collection of DSIs is straightforward.

6.7.5 Why We Want Reference Vectors

We get a special advantage when we have a set of Reference Feature Vectors (or simply, Reference Vectors) for our corresponding set of concepts. This is that we now have a mathematical means of specifying how close a DSI is to a certain concept. We can make mathematical combinations of these Reference Vectors to express “concept associations,” which could be a set of topics that would be discussed in various DSIs.

With this approach, we can then measure how close or far a given DSI is to different sets of concepts. We can use a variety of well-known clustering algorithms, among them being \textit{k-means} (and its neural network equivalent, the Learning Vector Quantization network, or LVQ). Thse algorithms require that we have a set of prototype vectors; vectors against which we match all the other “test” vectors.

Another way in which we could use these Reference Vectors is that they establish a sort of dimensional axis. Just as we position a point in \((x,y)\)
coordinate space by measuring its distance separately in the $x$ and $y$ dimensions, having Reference Vectors allows us to establish how much of various concepts are present in a given DSI. We can position a document in a multi-dimensional concept space. Our position in the direction of any concept $P_i$ is mathematically-based on the presence of contributing terms.

This is a step in the direction of dimensionality reduction; we are going from a very large number of terms to a (supposedly) smaller number of concepts. Even though we can have a very large ontology associated with a corpus, not all of the ontological nodes will be present, or will show up with much interest in our final model. Thus, going from terms to concepts is a good dimensionality reduction step.

A more well-formulated mathematical approach would be to do Principal Components Analysis (PCA), or a neural network equivalent. However, for our first efforts, we want to concentrate on direct term-to-concept mappings.

Of course, this is a highly simplified approach. However, it’s also a good starting point.

### 6.7.6 Defining the Reference Vector $R_i$

From the previous work, it seemed as though we would be able to specify the $R_i$ for concept $P_i$ as a simple vector composed of the term frequencies (tf’s) for each term relating to concept $i$. (Remember, there will also be lots of zeros in this vector, for the irrelevant terms.)

This approach has many advantages, chief among them being simplicity. However, we would still first have to identify which terms are relevant to our given concept, and give strengths - or use the $tf-idf$ - just for those, the rest being zero.

$$R_i = \begin{bmatrix} r_{i,1} \\ r_{i,2} \\ \vdots \\ r_{i,J} \end{bmatrix} \tag{6.2}$$

where

$$r_{i,j} = \begin{cases} tf_j, & \text{if } tf_j \text{ contributes to Concept } P_i \\ 0, & \text{otherwise.} \end{cases} \tag{6.3}$$
where

$$\text{tf}_j = \text{term frequency}_j$$  \hspace{1cm} (6.4)

We can express the previous equations using matrix algebra instead of a conditional statement. To do this, we have to multiply the $T$ term-frequency vector (or \textit{tf-idf} vector) by a weighting matrix $A$.

In the following equations, the suffix $i$ is dropped; we will assume that each matrix $A$ actually refers to $A_i$: the weighting matrix for concept $i$.

We need to use a square matrix, because we are multiplying each $j^{th}$ element in $T$ by its own assigned coefficient; recall that multiplying a $J \times J$ matrix by a vector (which is a $J \times 1$ matrix) gives us a resulting $J \times 1$ matrix, or vector.

Thus, the matrix element $a_{q,j}$ refers to the element in the $q^{th}$ row and $j^{th}$ column.

In the following equation, we have a diagonal matrix; only the matrix element $a_{j,j}$ multiplies the $j^{th}$ vector element. Essentially, each $j^{th}$ term frequency vector element is assigned its own coefficient; $a_{j,j}$.

$$R = \begin{bmatrix} a_{1,1} & 0 & \ldots & 0 \\ 0 & a_{2,2} & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & a_{J,J} \end{bmatrix} \begin{bmatrix} \text{tf}_1 \\ \text{tf}_2 \\ \vdots \\ \text{tf}_J \end{bmatrix}$$  \hspace{1cm} (6.5)

Note that all off-diagonal elements for this are zero, and that the diagonal elements themselves are zero only if that corresponding term contribute to the concept being defined by matrix $A$.

This is essentially giving us a \textit{prototype vector} for concept $P_i$. It tells us that if we were to have $P_i$ appearing in a certain DSI, that concept would be reflected by having the term frequencies for $P_i$ show up.
6.8 Thinking Through the Coefficient Matrix

The approach in the previous section was, of course, a bit simplistic. **We sometimes want to weight certain terms more than others.**

*Illustration:* Even if a term just shows up a little bit, we may want to weigh it strongly if its presence is a very strong indicator of a given concept. For example, in the *Benghazi-event*, a single mention of the term *Benghazi* is typically sufficient to confirm the massacre of four Americans at Benghazi, especially if the DSI is dated on or after Sept. 11, 2012.

Practically speaking, this means that we want values for the coefficients of matrix $A$ that are sometimes not either 0 or 1.

6.8.1 Drafting a Coefficient Matrix

Clearly, one of the biggest tasks in creating good concept definitions from RFVs is to carefully develop our coefficient matrix $A$.

We get one hint as to how this might be done from the logic behind the *tf-idf* (*term-frequency x inverse-document-frequency*). Since we already have the term-frequency in the vector, the *tf-idf* notion suggests that the coefficients $a_{ij}$ be set to the inverse-document-frequency for that term.

This isn’t a bad idea, at as a first step, we could certainly implement something like this.

We might also make a point of giving a very high weight (let’s say that our maximal weight is 1.0) to proper entity names, so that the extracted entity “Hillary Clinton” or even just “Clinton” (in the right context) could be given a weight of 1.0 for the *hillary_clinton* concept.

But this is still a bit simplistic. It will not give us a graded set of concepts for a given DSI; it will not necessarily smoothly map an ontological subspace (a portion of an ontology graph) to a DSI.

There is something else that we have to consider: the previous Eqn. 6.5 was a diagonal matrix; the resulting value for a given term in the RFV was just a single coefficient times the initial term-frequency for that term.

When we think about it, this is not very realistic.

It is much more likely that the presence of one term will occur in conjunction with a number of related terms. For example, in the *email_event*
associated with Hillary Clinton, the \textit{state\_department} concept (and all of its related terms) often co-occurred with a \textit{state\_department\_policies} concept. These two concepts (\textit{state\_department} and \textit{state\_department\_policies}) don’t always co-occur, but in the case of the \textit{email\_event}, they frequently did.

When the significance of one feature vector element is at least somewhat dependent on the presence or absence of other feature vector elements, we can still model it as a linear combination of feature vector elements. However, we no longer use a diagonal matrix. The coefficients that were assigned a value of 0 in Eqn. 6.5 will now sometimes take on non-zero values.

6.9 Assigning Coefficients (Strength Values): Examples

Let’s cap concept strength at 1.0. We want concepts to be represented at strengths between 0 and 1. (0 \leq C_i \leq 1.0)

This is not a complete example; it’s an excerpt of how this might be done. The next chapter will provide more details.

To create coefficient values for matrix $A$, we need to identify the strength - or relative importance - of a particular term in establishing the presence of a given concept.

6.9.1 Some Illustrative Rule Guidelines

\textbf{The only real rule is: We can make up our own rules.} What follows are illustrative for the kind of first-pass rules that you may choose to use in your application. You will undoubtedly tweak parameters and introduce refinements.

We don’t have to rely on \textit{tf-idf}, and in particular, we may want to make up our own rules to establish a consistent convention - and then adapt later with \textit{tf-idf}.

\textbf{Some Suggested Starting Rules:}

1. A person’s name - in any equivalent form - has strength 1.0 for the corresponding person instance. For example, “Hillary Clinton,” “Hillary,” “Clinton,” “Mrs. Clinton,” all are equivalent forms of the Hillary Clinton term. Thus, the Hillary Clinton term has a strength of 1.0 for the \textit{hillary\_clinton} instance.
2. A person’s role (in any equivalent form) has strength between 0.3 and 0.8 for the corresponding person instance. For example, the “Secretary of State” term has a strength of 0.8 for the hillary_clinton instance. It has a strength of 0.3 each for the condaleeza_rice and colin_powell instances, because they were further in the past. (We should also further define specific instances of these persons as being in-role as Secretary of State. (We can set up a table of former Secretaries of State, and other person in public office, scaling the concept association of their role to their instance of being in that office, as a function of time.)

3. When a role is played by a distinct small number of persons, divide the term-assignment-strength for the persons in that role into approximate fractions, where the fraction is about the number of times that DSIs in the corpus refer to a specific person or player, and the sum of the fractions is 1. For example, we have an ontology node hillary_clinton_board_member_clinton_foundation. The three most notable Board Members have been Hillary, Bill, and Chelsea. Approximate references to each in their role are about 0.5, 0.4, and 0.1. This is because recently, more DSIs have dealt with Hillary in that role (even though she’s stepped down) a bit more than Bill, and both more than Chelsea. The sum 0.5+0.4+0.1=1.0. Thus, we could assign a “Clinton Foundation” term strength of 0.5 to hillary_clinton_board_member_clinton_foundation.

4. When a role is inhabited by a somewhat larger distinct number of persons, assign a small initial fraction value to the strength of a person in that role. For example, Marco Rubio is a person-entity (marco_rubio) and is also in-role as a U.S. Senator (marco-rubio-senator). We have 100 Senators. We’ll give a strength of 0.1 to assigning the term “Senator” to the marco_rubio_senator entity, and 0.05 to the marco-rubio entity. This is more than 0.01, which we’d get if we divided 1 (the maximal term strength of “Senator”) by 100, but assigning too low a value to a term makes it less useful.

5. When we use a term to provide strength for an in-role instance, we will use a smaller strength to identify the entity without being in that role. For example, we could assign a strength of 0.5 to hillary_clinton_sec_state, and a smaller value of 0.3 to hillary_clinton. When we light up a match to a specific ontology node, we also want to reach a little bit to that node’s parent node.
Example: values for matching terms to concepts in the media sub-ontology:

<table>
<thead>
<tr>
<th>Term / Concept</th>
<th>media</th>
<th>tv</th>
<th>CBS News</th>
<th>newspaper</th>
<th>The New York Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>media</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>tv</td>
<td>0.2</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>cbs_news</td>
<td>0.1</td>
<td>0.5</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>newspaper</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>new_york_times</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.5</td>
<td>1.0</td>
</tr>
</tbody>
</table>

In this example, each term supports its direct concept at a strength of 1.0. (The term "New York Times" supports the concept new-york-times at a strength of 1.0.) "New York Times" supports the parent node newspaper at a smaller strength of 0.7, and the grandparent node at a still smaller strength of 0.5. We want to invoke the parent and grandparents as concepts associated with this DSI, but they’re either not as strong - or will get more broad activation from OTHER specific instances.

For example, the mention of "New York Times" will only activate the media node with a strength of 0.5. But if the same article also has the term "CBS News", and that activates the media node with a strength of 0.5, then we have strong overall activation of the higher-level media node as well as the specific instance nodes

6.9.2 A First-Pass Example: Two Entity Instances

The example here focuses on Hillary Clinton (specifically, the concept hillary_clinton) as an instance of a candidate for the presidential election of 2016. We are also interested in a further instance; Hillary Clinton in her role as Secretary of State during the first four years of Obama’s presidency.

Concept: hillary_clinton
<table>
<thead>
<tr>
<th>Term</th>
<th>term count</th>
<th>tf</th>
<th>concept-mapping-strength</th>
<th>concept-strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clinton*</td>
<td>11</td>
<td>0.1</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Secretary of State</td>
<td>2</td>
<td>0.02</td>
<td>0.3</td>
<td>0.006</td>
</tr>
<tr>
<td>potential Democratic candidate for 2016 presidential election*</td>
<td>1</td>
<td>0.01</td>
<td>0.2</td>
<td>0.002</td>
</tr>
<tr>
<td>Clinton Foundation</td>
<td>0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>email*</td>
<td>7</td>
<td>0.07</td>
<td>0.3 (est)</td>
<td>0.021</td>
</tr>
<tr>
<td>personal email address*</td>
<td>3</td>
<td>0.03</td>
<td>0.3 (est)</td>
<td>0.009</td>
</tr>
<tr>
<td>Benghazi*</td>
<td>1</td>
<td>0.01</td>
<td>0.2 (est)</td>
<td>0.002</td>
</tr>
<tr>
<td>Summed Concept Strength</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.14</td>
</tr>
</tbody>
</table>

* term is an equivalence class; several terms can contribute to this term-count

**Concept: Hillary Clinton in-role as Secretary of State**

(*hillary_clinton_sec_state*)

The previous table introduced a couple of the terms that match towards this sub-node. Notice how the concept mapping-strength has changed, e.g. for the relevance of the term "Secretary of State" to Hillary Clinton in-role as Secretary of State.
<table>
<thead>
<tr>
<th>Term</th>
<th>term count</th>
<th>tf</th>
<th>concept-mapping-strength</th>
<th>concept-strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clinton*</td>
<td>11</td>
<td>0.1</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Secretary of State</td>
<td>2</td>
<td>0.02</td>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>State Department*</td>
<td>1</td>
<td>0.01</td>
<td>0.2</td>
<td>0.002</td>
</tr>
<tr>
<td>service at State</td>
<td>1</td>
<td>0.01</td>
<td>0.2</td>
<td>0.002</td>
</tr>
<tr>
<td>email*</td>
<td>7</td>
<td>0.07</td>
<td>0.4 (est)</td>
<td>0.0281</td>
</tr>
<tr>
<td>personal email address*</td>
<td>3</td>
<td>0.05</td>
<td>0.5 (est)</td>
<td>0.015</td>
</tr>
<tr>
<td>Benghazi*</td>
<td>1</td>
<td>0.01</td>
<td>0.3 (est)</td>
<td>0.003</td>
</tr>
</tbody>
</table>

* term is an equivalence class; several terms can contribute to this term-count.

So far, we have two concepts: *hillary_clinton* with summed-concept-strength = 0.14 and *hillary_clinton_sec_state* at 0.16.

It’s not surprising that the child node (*hillary_clinton_sec_state*) is a bit stronger than the parent (*hillary_clinton*); when we read the originating DSI, it’s not about Clinton overall; it’s about her actions while she was *in-role* as Secretary of State.

One thing we want to do is to scale concepts for each DSI. There are often a LOT of terms, so simply using the term-frequency is not the right scale.

One approach is to map the strongest concept in a given DSI to a value of 1.

Our strongest concept so far is *hillary_clinton_sec_state*, with a summed-concept-strength of 0.16.

\[ 0.16 \times 6.25 = 1.0 \]

Our new scaling factor is 6.25. It yields: *hillary_clinton*: 1 *hillary_clinton*: 0.875.
We could wind up changing the scaling factor when the strengths of other concepts are computed, such as email. This will be addressed in a later chapter.

6.10 Next Steps: Identifying Novelty

Even though the preceding section identifies an initial algorithm for determining how terms can be used to define a concept, there is still a lot of ambiguity. This is because one term can be used to define several concepts.

Thus, when a given term \( j \) shows up in a DSI, we don’t have - at this moment - a means to parse the extent to which term \( j \) expresses concept \( k \) or concept \( k+1 \), assuming that term \( j \) references each of those concepts.

Leaving aside that difficulty for the moment (but keeping in mind that we would have to invoke a range of methods, from Principal Components Analysis to Independent Components Analysis), let’s assume that we’ve identified all the concepts that are present in a given DSI.

Suppose that there are a some terms left over; terms that are not matched against pre-existing concepts.

We would then want to create a bucket list of all these left-over terms, and see if there is something new that is showing up in a significant way - perhaps these terms are occurring in a lot of DSIs that happen on or after a certain date.

This is what would have happened if a text analytics system was processing new DSIs about Hillary Clinton starting in early March, 2015. Prior to this time, the term email was rarely - if ever - associated with the concept Hillary Clinton. Beginning in early March, the term email (and many related terms) began to show up in DSIs that had the Hillary Clinton concept.

This would be the time and place for novelty detection.

The first and crudest method would simply be to run a \( \text{tf-idf} \) process and extract those new terms that were not already part of a pre-defined concept and ontology set. If they exceeded some threshold, they could trigger an alert for corpus examination, and the whole process of ontology definition, concept association to ontologies, and concept definition would begin all over.

This time, though, we would start to see co-occurrence of the new set of concepts with the pre-existing Hillary Clinton concepts; a co-occurrence matrix would reveal some interesting things.
Bibliography

[1] IBM SPSS Text Analytics, *Mastering new challenges in text analytics: Making unstructured data ready for predictive analytics*. Use the title as Google keywords to access and download the PDF.