TopicWeb: A Novel Approach to Automatic Document Similarity Measurement and Categorization

Stanislav Nikolov

1

1.1 A little bit about my project

The project introduces the TopicWeb, a system for improved similarity measurement between two textual documents (news articles, emails, etc). It consists of an interconnected web of documents, where a connection between two documents specifies the strength of the similarity between the topics of those documents. A standard method of calculating similarity between two documents is improved upon by examining not only the two documents themselves, but also their neighbors in the TopicWeb. This method effectively differentiates between documents with topics that may be similar sounding to a computer - like "heart surgery" and "Valentine's day heart" - but are fundamentally different.

1.2 How did I get the idea for my research?

To find the right topic for a research project, one must strike the right balance between not hesitating to jump into a particular topic, and being open to new ideas if that topic doesn't work out. Make a list of all the fields that would be exciting for you to do research in and try to find specific problems to be solved in each field. My list ranged from face recognition to automatic summarization to artificially intelligent control systems for spacecraft. It's important not to feel stuck with a certain field because you've already done weeks or even months of searching for a topic within that field, and starting anew would make you fall behind. If nothing interesting or feasible comes up, then try switching your topic. However, don't feel discouraged if you don't initially understand the material you read - you'll have plenty of time to understand it if you eventually choose the topic. The main motivation should be whether something is interesting and exciting to you.

I eventually picked the topic of automatic summarization of documents, which involves creating a shortened version of a document by analyzing the most important topics, or themes, in the document. The more I read about the field, the more interesting and feasible it seemed. Eventually, I got interested more specifically in the identification of topics in a document and the measurement of how topically related two documents are. At the beginning of last summer, I set out to try to improve one of the standard methods of measuring such similarity.

1.3 Where and with whom did I do the research?

The research was done during a summer internship at Morgan Stanley, in the Information Technology department, where I got permission to work on my project independently. I decided I would implement a standard method for document similarity measurement (the Vector Space Model, described later on), see what was wrong with it, and try to improve it. My mentor — although he was not a traditional research mentor, such as a scientist or an expert in the field — helped me narrow down my topic and get my summer internship. I did not have to learn much math for the project — I was already familiar with linear algebra and graph theory. But in retrospect, having more knowledge of math and algorithms would've helped me create a more successful model.

1.4 How the research changed me

The research definitely made me a more confident problem solver because I managed to wrap my head at least partially around a real problem that does not have a nice, easy answer. I look back now at how I gained this confidence last summer. At the outset of solving a problem, a finding a solution, or even coming up with an idea, seemed like an insurmountable task. By the time I fully got into the problem however, I was overwhelmed with so many ideas that I could hardly manage to write them all down. But that is exactly what I did - I wrote them all down. Knowing that I had all my options before me freed me to jump in and explore a single idea in sufficient depth and enabled me to backtrack if I reached a dead end. There was also a certain comfort in

knowing which theories, methods, or experiments worked at any given point. This allowed me to take things one step at a time - to add on to the list of ideas and theories that have been validated, and to form new ideas and theories based on the previous ones.

For example, I started my project with only the vaguely defined goal of improving the accuracy of automatic document similarity measurement. By questioning one aspect of an existing algorithms, I would come up with an alternative, which would invariably have its own deficiencies, the analysis of which would often shed new light on the problem. Through this unpredictable and winding path, I developed a series of algorithm that are simply the conglomeration of all the individual ideas that evolved out of this process and made logical sense.

And I sincerely hope that by the time you finish reading this article it mostly makes sense, and even sparks your interest enough to try to improve upon a project that I can personally say has many deficiencies.

2

2.1 Introduction

In an age of information overload on the Internet and in various professional settings, it is not unusual for someone to spend valuable time sifting through useless data. This is a major problem especially in large corporations, where intranets contain tens of thousands of files and many people receive hundreds

or even thousands of e-mails a day. The process of manually determining what is important and what is not not only wastes time that could be better spent, but also might result in human error, whereby important data could go unread.

This situation calls for automation of a method that provides increased accuracy in identifying the topic of a document — a textual pice of data like an email or an article — and in calculating its similarity to other documents.

I will first describe a standard method for calculating such similarities and then describe what I have done to try to improve upon it.

2.2 Vector Space Model for document similarity measurement

So how do we calculate how similar two documents are? One of the classic ways to do this uses what is called the Vector Space Model, as developed by Salton, Wong, and Yang in their paper "A Vector Space Model for Automatic Indexing."

2.2.1 A Cloud of Points

To understand the vector space model, imagine a cloud of objects in space with certain defining characteristics. For now, let those defining characteristics be the x-coordinate, the y-coordinate, and the z-coordinate of each object. Thus, the objects are drawn as points in relation to the x, y and z axes.

If we are given one of these points, how do we find the "closest" point? This depends on the definition of closeness. For now, let the closeness be defined as

the distance between the two points. Thus, in order to find the closest point, we simply compute the distance from our given point to each other point and take the point with the shortest distance.

Now, let's define another measure of closeness. First, however, we have to make a minor change. Each of our points has a certain distance from the origin, and this distance is different for every point. Suppose we want to make this distance the same for all points — say a distance of 1. We can do that by taking the line connecting each point to the origin, and moving the point along that line until the distance is 1. If we do this for every point, our points will lie on the surface of a sphere. Now, we can define a new measure of closeness. For any two points, we can take the lines from the origin to each point and measure the angle between the two lines. The smaller the angle, the closer the two points are.

This latter method will prove more useful.

2.2.2 So what does this have to do with documents?

It turns out extending this abstract cloud model to make it compare documents to each other isn't too difficult. The characteristics of each object are just coordinates that place it somewhere in *physical* space — the space that we inhabit.

What if we had a set of four characteristics for each object? And what if we called them weird names like door, computer, bottle, and poster? We can define an object in this fictitious 4-dimensional space by assigning to it

numerical values for door, computer, bottle and poster, just like we placed an object in 3-dimensional space by assigning to it values for x, y and z. Of course, we can no longer visualize something that exists in four dimensions, but for our purpose, we don't need to visualize it.

Now, we are ready to extend this model to the world of documents. To do that, we need to find out just what characteristics define a document. Perhaps the most natural answer is to go through the documents and count how many times each word appears.

The first instinct one might have is to rank the words in descending order according to how often they occur. Those at the top should seemingly be most important and most representative of the document. For example, the Wikipedia article about palm trees should have many occurrences of "palm," "tree," "leaves," etc. But what about words like "the" and "a"? These words appear very often but they don't tell us anything special about a specific document. Why not? Because they are present in abundance in most other documents written in English and thus, they do not help us to differentiate one document from another.

Keeping these things in mind, we need to find a way to assign a value to each word in each document that measures that word's importance, or *weight*.

Later on, we'll see how we can use all the weights in a document as the set of characteristics that place the document somewhere in multidimensional space, but first we need to see how to calculate those weights.

2.3 Computing weights

Now we are ready to compute the weights of all the words in a document.

First, we take all the words in a document and use an algorithm to strip their suffixes, plurals, etc, if they have any. The words "program," "programs," and "programming" all give essentially the same meaning — that the document is about something related to programs and programming. Then, we count how many of each word there are in the document. For each word, we take the number of times it occurs and divide it by the total number of words — this is called *term frequency*.

Next, we need to take into consideration the fact that a document is part of a collection of documents. In the previous section, I mentioned that words like "the" and "a" are meaningless because they appear in pretty much every English document and fail to differentiate one document from another. This issue of being able to differentiate between documents requires the existence of a collection of documents between which we can differentiate.

So how would we ensure that the weights for such meaningless words are in fact low, the way they should be? The most common way is, for each word, to count the number of documents in the collection that it appears in at least once. This value is called the *document frequency*. If this number is high, it can be used to offset the term frequency to a lower value, which is what we want.

We can define the term frequency and the document frequency more formally now and finally come up with a formula to calculate weights.

The term frequency of term t_i is given by $tf_i = n_i \Big/ \sum_{k=1}^{m} n_k$, where n_i represents the number of occurrences of term i in the document, and m is the number distinct, or different, words in the document.

The document frequency of term t_i is given by $df_i = |\{d : d \ni t_i\}|/|D|$, where $|\{d : d \ni t_i\}|$ is the size of the set of all documents that contain term t_i and $|D|$ is the size of the set of all documents.

In our case, it is more useful to define the *inverse document frequency*, which is $idf_i = log\frac{1}{df_i} = log[|D|/|\{d : d \ni t_i\}|]$

We can combine the term frequency and the inverse document frequency as follows to give the weight of term t_i in the document d:

$$w_{t_i,d} = (tf)(idf)$$

2.4 Creating representations of documents based on their weights

In this section, I will describe how we can use the weights of the words in each document to create a mathematical representation of that document. When we looked at points in three-dimensional space, they each had the exact same set of characteristics consisting of an x-coordinate, a y-coordinate, and a z-coordinate. Some points might lie on the xy plane (no z component) but we still give them a z value of 0 because we need the same set of characteristics and that includes having a z-coordinate.

Similarly, we need the exact same set of characteristics for each document

in a collection of documents. As you have probably already guessed, these characteristics are the words in the documents. However, not all documents have the same words, and we need a set of characteristics that applies to all the documents. To do that, we simply take the set of all distinct words that appear in the entire collection of documents. This is an ordered set of characteristics, the same way that (x, y, z) is an ordered set of characteristics. An ordered set of n characteristics that each have numerical values can be thought of as a vector in n-dimensional space.

Now, for each document, each member of a set of characteristics needs to be assigned a value. Since each characteristic is a word, and we already know the weight of each word in each document, we do one of two things. If that word is in the document, we just assign it the value of the weight of that word in the document. If the word is not in the document, we assign it a value of 0.

Now we can represent all the documents in our collections as vectors in a multidimensional space.

2.5 Computing similarities

Earlier, we discussed two methods of calculating similarity between objects in three-dimensional space — using the distance formula on the two points in space, or making their distance to the origin equal and calculating the angle between them. These methods work exactly the same way in 4-dimensional space, in 5-dimensional space, or in a space of any dimension, because we can always find the distance between two points, regardless of their dimension, and an angle

always exists between two vectors regardless of their dimension.

I also mentioned earlier that the angle method was more useful for our purpose and I will now introduce it formally and explain it in more detail.

First, we need to make all the vectors have length 1. We can do this to a vector by normalizing it, or dividing it by its magnitude.

Second, we need to find some measure of the similarity involving the angle between the two vectors. There is a very convenient operation that one can do to two vectors that gives us an expression that involves the angle between the two vectors. The operation is called the dot product, defined as

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n,$$

where $\mathbf{u}$ and $\mathbf{v}$ are n dimensional vectors and u_i, v_i are the ith components of vectors $\mathbf{u}$ and $\mathbf{v}$, respectively.

It turns out that

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}| \cos\theta(\mathbf{u}, \mathbf{v}),$$

where $\theta(\mathbf{u}, \mathbf{v})$ is the angle between vectors $\mathbf{u}$ and $\mathbf{v}$, and $|\mathbf{u}|$ is the magnitude of vector $\mathbf{u}$.

Since the vectors are normalized, their magnitudes are equal to 1. Thus, if the dot product is 1, the angle between the vectors must be 0, indicating maximum similarity. If the dot product is 0, then the angle is 90 degrees, indicating no similarity.

The reason the vectors are normalized is that the vectors of long documents will have large magnitudes simply because they contain more words. We can

see from the above formulas that vector magnitude will affect similarity. Since length has nothing to do with the meaning, or topic of an article, we want to not consider it and we do this by making the magnitudes of all vectors 1.

In the next few sections, I will talk about the work that I have done to try to improve the Vector Space Model.

2.6 Motivation for TopicWeb

Although dot product similarity is a very clever and useful technique, it has a characteristic flaw, which should become clear with the following example.

Consider the vectors of two documents, one about the city of Phoenix, and another about the phoenix as a mythical bird. Both documents will likely contain the term "phoenix" many times and the tf-idf weight of "phoenix" will be very high in both. Through observation, the dot product similarity in such cases is almost as high, and sometimes higher than the similarity between documents of the same topic (ex. two documents about phoenix mythology or two documents about Phoenix, Arizona). Thus, it is necessary to develop a better similarity metric that involves an improved representation of the actual topic of each document, one that will reduce such ambiguity.

2.7 Topical similarity

In the previous section, we saw that most of the similarity came from one prominent word that the two documents had in common. Thus, a logical approach is to reward with a higher similarity those pairs of documents that have a higher

number of highly weighted terms in common. An important question is, how highly weighted is high enough? Another important question is, how much do we increase the similarity based on how many of these terms there are in common?

Let's call the set of *relevant terms* the set of terms in the document that have the N highest weights. Note that several terms can have the same weight, resulting in the set of relevant terms being larger than N, and that the document may not have N different terms, resulting in a set of relevant terms being smaller than N.

We can compute the number of matches between the sets of relevant terms of two documents, or in other words, the size of the intersection set. Next, since our dot product similarity ranges from 0 to 1, it makes sense to convert the number of matches to a value from 0 to 1 as well and multiply the two values. To do this, we can calculate the size of the intersection of the two sets of relevant terms and divide by the average size of the two sets. This value will have a minimum of 0 when there are no matches, and a maximum of 1 when the sets are of the same size and contain the same terms. Let's call this value the *term match density* ρ, given by

$$\rho(R_a, R_b) = 2\frac{|R_a \cap R_b|}{|R_a| + |R_b|}$$

where R_a and R_b are the sets of relevant terms of documents a and b, respectively. Using ρ, we can define the *topical similarity* σ which we get by multiplying the dot product similarity by the term match density. This is given by

$$\sigma(a, b) = \rho(R_a, R_b)(v_a \cdot v_b)$$

where v_a and v_b are the vectors of documents a and b, respectively.

2.8 TopicWeb model

We can use dot product similarity and topical similarity to create a graph of documents. A graph is a set of nodes, or objects that have edges connecting some or all of the objects to each other. We want to represent the relationships of documents to each other and such a web of relationships is fitting.

We say that an edge exists for all pairs of documents whose dot product similarity exceeds a certain threshold τ. Topical similarity is then used to define the strength of the relationship between the topics of the two documents.

2.9 So what exactly can we do with the TopicWeb?: Term Diffusion algorithms

I have come up with an algorithm to analyze the TopicWeb in order to allow a document to be influenced by neighboring documents. This influence is in the form of the "diffusion" of terms across documents that are closely linked in the TopicWeb. Consider a group of close friends, and a specific person in the group. You could ask that person about their interests, but the answer could be unreliable — the person could leave things out, or talk about things they're not that interested in but perhaps happened to pop into their heads. If the group of friends is very close and has very similar interests, you could also ask that

person's friends what their own interests are. This is exactly what we're going to do with documents.

Given a document u and its relevant term set R_u, its neighbors n_1, n_2, n_3,...,n_m, and their respective relevant term sets R_{n_1}, R_{n_2}, R_{n_3},...,R_{n_m}:

1. Let $D_i \subset R_i$. D_i is the set of terms from a neighboring document that are "allowed" to diffuse into the central document.

2. For each relevant term set R_{n_i} and for each term t_j in D_{n_i}: (a) calculate the following value: the weight of t_j in document n_i times the square of the topical similarity (inverse of the edge length in the TopicWeb) σ times a constant $\kappa \geq 1$ and (b) add this value to the weight of t_j in u. κ will be used as a variable to regulate the amount of diffusion, or the extent to which neighboring documents can influence the central document.

3. Normalize the vector of document u to get a unit vector direction.

This algorithm changes the components of the document vector and results in a new unit vector representation of the document's topic. The unit vectors that result from this algorithm can be treated in the same way as individual document vectors, and compared to each other using dot product similarity.

Compared to the original vectors of the documents, their modified vectors should be closer together if the documents are of the same topic, and farther away if the documents are of different topics.

The TopicWeb model itself (without the term diffusion algorithms) is very simple yet fundamental because it represents relationships in a graph. There are

probably many things that one could do with it, perhaps by using techniques that are commonly used with graphs, or something completely different. Admittedly, I haven't thought much about that. I just took one of the first ideas I had for a way to use this model and ran with it. I challenge you, the reader, to come up with some other useful algorithm to handle and analyze these relationships.

In the context of the algorithmic description, we will define two different types of diffusion. Simple Term Diffusion is the specific case where $D_i = R_i \ \forall \ i$, and Selective Term Diffusion, is the case where D_i is equal to R_i minus the term with the highest weight in R_i.

But first, I'll talk about the types of documents that I tested this on. The document collection consisted of a subset of the Reuters 21578 Document Corpus and a smaller, manually added test collection. 5081 articles with character count exceeding 1000 were chosen from approximately 13,000 articles in the Reuters Corpus. The large size of the collection is important to guarantee statistically sound term weights over a fairly heterogeneous spectrum of topics.

Fifteen carefully selected documents were manually added to the original 5081 — 10 articles about the mythology of the phoenix, and 5 about Phoenix, Arizona. These will serve as the case study for the results of experimentation with the TopicWeb. This set of documents will be referred to as the **Phoenix Corpus**.

The results and their explanations are in Figures 1 and 2.

As you can see, Simple Term Diffusion doesn't do a great job at differentiating between documents of different topics, while Selective Term Diffusion is

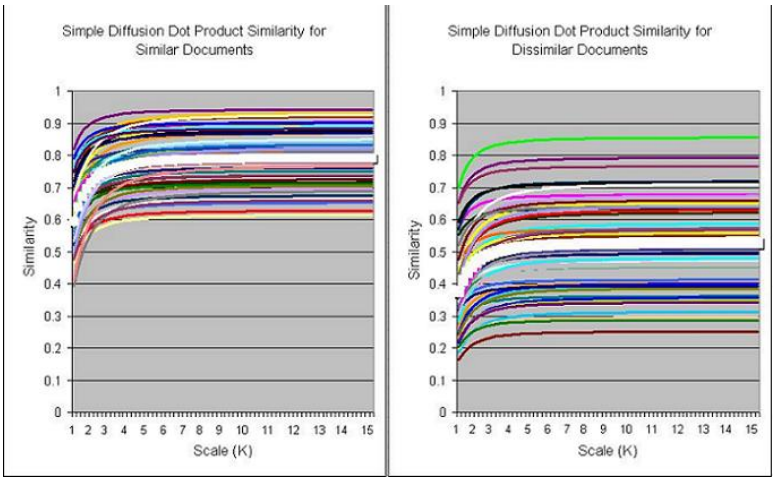

Figure 1: Left: Simple Diffusion Dot Product Similarity for documents of the same topic. Right: Simple Diffusion Dot Product Similarity for documents of different topics with one or more ambiguous words that cause deceptively high similarities. The data was collected from tests on the Phoenix Corpus.

more successful.

Since the term "phoenix" is arguably the most common term in the entire Phoenix Corpus, its value in any document vector will increase in proportion with increased term diffusion. Since this applies to documents from both categories of Phoenix Corpus articles, the term "phoenix" will quickly become the dominant term in almost all document vectors, thus increasing the dot product for almost all pairs of documents being compared, regardless of whether they are similar or dissimilar. In contrast, Selective Term Diffusion The limits the dominance of ambiguous terms like "phoenix" and disables them from undergoing diffusion. By not allowing such terms to diffuse into a given document vector, their weights in that vector will drop, while other terms that do diffuse will increase in weight. When a single term no longer dominates, other terms that are characteristic of a document's topic gain a newfound importance.

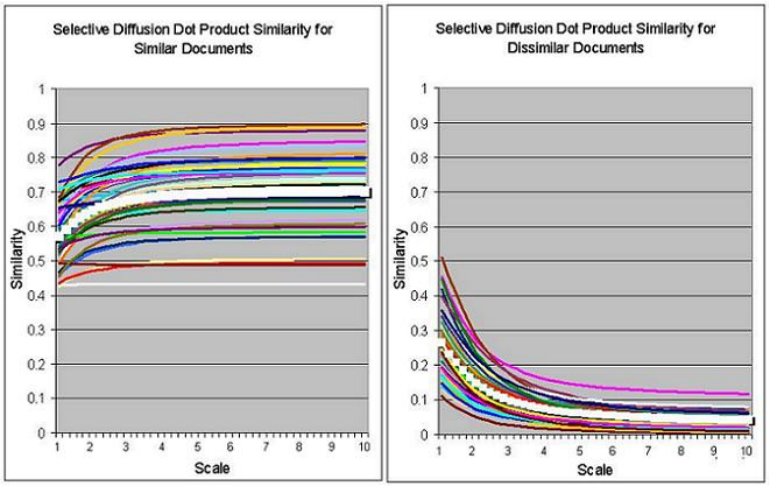

Figure 2: Left: Selective Diffusion Dot Product Similarity for documents of the same topic. Right: Selective Diffusion Dot Product Similarity for documents of different topics with one or more ambiguous words that cause deceptively high similarities. Selective Diffusion clearly performs better than Simple Diffusion in differentiating between documents of different topics. The data was collected from tests on the Phoenix Corpus.

2.10 Hierarchical clustering: Dot Product Similarity vs. Selective Diffusion Dot Product Similarity

Hierarchical clustering is a method used to organize data in a hierarchy using a binary tree. The basic operation for hierarchical clustering is similarity calculations — so the better the similarity metric, the better the clustering should be[1]. Figure 3 shows the results of performing hierarchical clustering on the Phoenix Corpus with Selective Diffusion Dot Product Similarity and with just Dot Product Similarity. Selective Diffusion Dot Product Similarity is clearly the better similarity metric for this purpose.

[1]To find out how this is done in more detail, visit http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/AppletH.html.

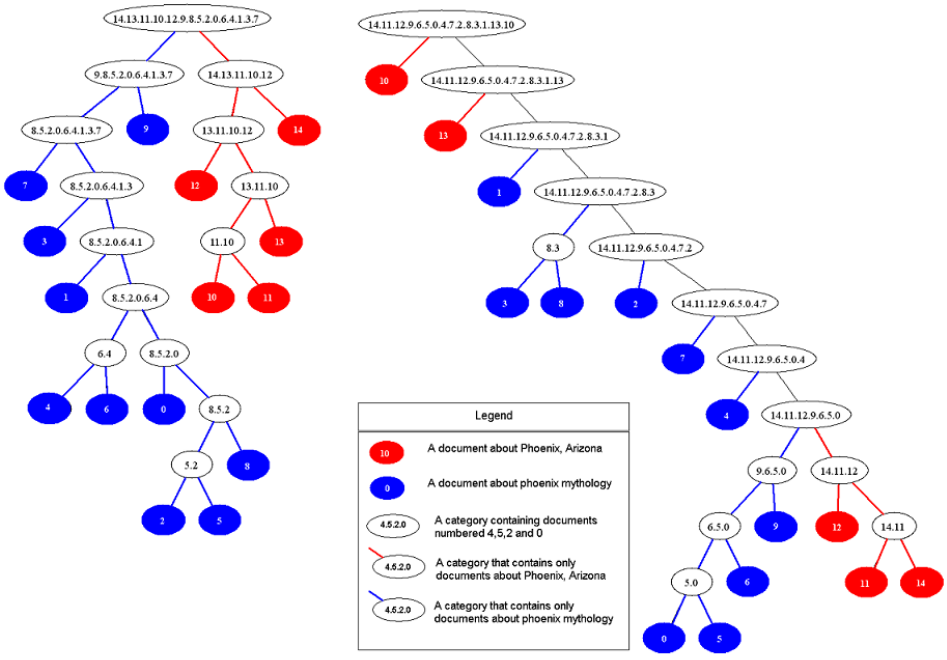

Figure 3: The documents about phoenix mythology (blue) and Phoenix, Arizona (red), after undergoing hierarchical clustering. The white, numbered ovals are categories, which contain at least two documents. **Left:** Hierarchical Clustering with Selective Diffusion Dot Product Similarity: The results show a clear hierarchical organization of topics starting from the topmost node. This results in logical visual separation of nodes belonging to different topics and connectedness of nodes belonging to the same topic. This hierarchical tree is easy to navigate and useful in searching through the collection of documents. **Right:** Hierarchical Clustering with Dot Product Similarity: Dot product similarity is generally a poor similarity metric for hierarchical clustering, as these results show. There is a lack of appropriate organization, most notably a lack of clear separation of nodes belonging to different topics as well as undesirable connectedness of nodes belonging to different topics. Because of these flaws, this tree is difficult to navigate and not very useful for searching through the collection of documents.

2.11 Conclusion

In this paper, I have proposed the TopicWeb Model for document similarity measurement and categorization. The initial problem that I aimed to resolve with this model was the problem of topical ambiguity in the traditional Vector Space Model. The goal of the experiments performed was to create a similarity metric that differentiates between the two different categories of topics that share an ambiguous term. Our case study was the Phoenix Corpus, where we examined "phoenix" related articles. Two different methods of utilizing inter-document relationships in the TopicWeb were presented - Simple Term Diffusion and Selective Term Diffusion. Selective Term Diffusion combined with dot product similarity proved more useful than dot product similarity alone in correctly differentiating between documents, and in organizing them by topic through hierarchical clustering.