UNIT 9 MINING BIG DATA

Structure **Page No.**

9.0 INTRODUCTION

In the previous Block, you have gone through the concepts of big data and Big data handling frameworks. These concepts include the distributed file system, MapReduce and other similar architectures. This Block focuses on some of the techniques that can be used to find useful information from big data.

This Unit focuses on the issue of finding similar item sets in big data. The Unit also defines the measures for finding the distances between two data objects. Some of these techniques include Jaccard distance, Hamming's distance etc. In addition, the Unit also discusses finding similarities among the documents using shingles. Finally, this unit introduces you to some of the other techniques that can be used for analysing Big data.

9.1 OBJECTIVES

After going through this Unit, you will be able to:

- Explain different techniques for finding similar items
- Explain the process of the collaborative filtering process
- Use shingling to find similar documents
- Explain various techniques to measure the distance between two objects
- Define supervised and unsupervised learning

9.2 FINDING SIMILAR ITEMS

Finding similar items for a small set of documents may be a simple problem, but how do you find a set of similar items when the number of items is extremely large? This section defines the basic issues of finding similar items and their applications.

Problem Definition:
Given an extremely large collection of item sets, which may have millions or billions of sets, how to find the set of similar item sets without comparing all the possible combinations of item sets, using the notion that similar item sets may have many common sub-sets.

Purpose of Finding Similar Items:

1. You may like to classify web pages that are using similar words. This information can be used to classify web pages.
2. You may like to find the purchases and feedback of customers on similar items to classify them into similar groups, leading to making recommendations for purchases for these customers.
3. Another interesting use of similar item set is in entity resolution, where you need to find out if it is the same person across different applications like e-Commerce web site, social media, searches etc.
4. Two or more web pages amongst millions of web pages may be identical. These web pages may be plagiarized or a mirror of a simple website.

Why Finding Similar Items is a problem?

One of the simplest ways to find similar items would be to compare two documents or web pages and determine if they are identical or not by comparing sequences of characters/words used in those documents/web pages. However, considering that you need to find duplicates in 10 documents/web pages, you need to compare ${}^{10}C_2 = 45$ pairs. In general, for n documents/webpages, you may need to compare $n \times (n-1)/2$ pairs of documents/web pages. For about 10^6 documents/web pages, you need to make about 5×10^{11} comparisons. Thus, the question is how to find these identical documents/web pages amongst billion of documents/web pages without checking all the combinations.

Next, we first discuss a technique to find the similarity of sets, followed by techniques that can be used to find similar item sets efficiently.

9.3 FINDING SIMILAR SETS

In this section, we discuss one of the measures of finding similarity among the sets and then discuss how this similarity measure can be used for finding the textual similarity of documents; and collaborative filtering, which can be used for finding a similar group of customers.

9.3.1 Jaccard Similarity of Sets

Jaccard similarity is defined in the context of sets. Consider two sets – set A and set B, then the Jaccard similarity $JS_{A,B}$ is defined as a ratio of the cardinality of the set A∩B and cardinality of the set A∪B. The value of $JS_{A,B}$ can vary from 0 to 1. The following equation represents the Jaccard similarity:

$$JS_{A,B} = \frac{|A \cap B|}{|A \cup B|}$$

For example, consider the sets A={a, b, c, d, e} and set B={c, d, e, f, g}, then Jaccard similarity of these two sets would be:

$$JS_{A,B} = \frac{|\{c,d,e\}|}{|a,b,c,d,e,f,g|} = \frac{3}{7}$$

9.3.2 Documents Similarity

Finding document similarity is an interesting domain of use of Jaccard similarity. It can be used to identify a collection of almost similar documents, news items, web pages or reports. This is also referred to as the character-level similarity of documents. Another kind of document similarity also looks at documents having similar meanings. This kind of similarity requires you to identify similar words and sometimes the meaning of the words. This is also a very useful similarity, but it can be solved using different types of techniques. How can you identify, if two documents are identical? This can be done simply by character-by-character matching. However, this algorithm may be of little use as many documents may be mostly similar though not exactly. The following cases specify this situation:

Plagiarism Checking: The plagiarized text may differ in small segments, as some portions of the document may have been changed or even the ordering of some of the sentences may be changed.

Mirror Websites: Even the mirror websites also make changes, as per the local advertisements and requirements.

The versions of Course Material: Even the versions of older course material and newer course materials, assignments etc. available on different websites may be slightly different.

Similar News Reports: The news reports about a news item may consist of similar text, which may be appended with supplementary information by each newspaper.

9.3.3 Collaborative Filtering and Set Similarity

In the past decade, e-commerce has gained acceptance by customers. Many of you, who have purchased and have liked various items on these e-commerce websites, get online recommendations for the purchase of certain items. You may have observed that many times those recommendations are very close to purchases that you would like to do in near future. This is achieved by the process of collaborative filtering, which tries to group you based on your purchases and likes with other users making similar purchases and likes and then recommending you the products that another person in the group has purchased and liked.

Similarity of the sets is used to address this problem. Two customers can be in a similar customer group if they purchase and like similar products. This similarity is determined by the Jaccard set similarity. However, please note that the number of customers, as well as the products that can be purchased on an e-commerce website, are very large. For example, two separate customers may be interested in purchasing books. They may purchase the same books available on Data Science and may rate them similarly. Please note they may also purchase many other books, which may be bought by only one of them. How are the problems of collaborative filtering and document similarity are different? The prime difference is in the value of Jaccard similarity. For document similarity, you are looking at $JS_{A,B}$ in the range of 50% or above, whereas for the case of collaborative filtering, you may be interested if the value of $JS_{A,B}$ is as low as 10-25%.

In general, collaborative filtering uses binary data, such as like/dislike or seen/not seen or purchased/not purchased etc. This data can easily be translated into a set such as a set of likes of items; a set of purchased items; etc. Thus, finding Jaccard's similarity is straightforward. However, in many situations, a 5-, 7- or 11-point Likert scale may be used to rate the articles. An interesting way to deal with such data may be to create a bag instead of a set having as many instances of an item, as the rating. For example, consider that on a 5-point scale, a customer gave a rating of 2 to product A and a rating of 3 for product B and another customer gave a rating of 1 to product A and a rating of 4 for product B, then the related data would be represented as:

BagCust1 = {a, a, b, b, b}
BagCust2 = {a, b, b, b, b}

$$JS_{A,B} = \frac{|\{a, b, b, b\}|}{|a, a, b, b, b, b|} = \frac{4}{6} = \frac{2}{3}$$

Thus, the set similarity is an important consideration for finding the similarity between two sets. In the next section, we discuss the method to convert a document into subsets of very small strings, which can be used to compute the similarity of two documents.

Check Your Progress 1:

Question 1: What is the major problem in finding similar items?
Question 2: Define Jaccard Similarity with the help of an example.
Question 3: What is collaborative filtering? How can it be addressed using set similarity?

9.4 FINDING SIMILAR DOCUMENTS

In the last section, we discussed the Jaccard similarity in the context of sets. We also explained the issues of document similarity. The document similarity can be checked using the following process:

Step 1: Create Sets of items from the document (Shingles are used)
Step 2: Perform Minhashing and create documents Signatures that can be tested for similarity checking. This reduces the number of shingles to be tested for checking the similarity of two documents.
Step 3: Perform Locality Sense Hashing produces pairs of documents that should be checked for similarity rather than all the possible pairs

In this section, we discuss these three important concepts of document similarity analysis.

9.4.1 Shingles

In order to define the term shingle, let us first try to answer the question: How to represent a document as a set of items so that you can find lexicographically similar documents?

One way would be to identify words in the document. However, the identification of words itself is a time-consuming problem and would be more useful if you are trying to find the semantics of sentences. One of the simplest and efficient ways would be to divide the document into smaller substrings of characters, say of size 3 to 7. The advantage of this division is that for almost common sentences many of these substrings would match despite small changes in those sentences or changes in the ordering of sentences.

A k-shingle is defined as any substring of the document of size k. A document has many shingles, which may occur at least once. For example, consider a document that consists of the following string:

"bit-by-bit"

Assuming the value of k=3 and even using a blank character as part of substrings. The following are the possible substrings of this document of size 3:

"bit", "it-", "t-b", "-by", "by-", "y-b", "-bi", "bit"

Please note that out of these substrings "bit" is occurring twice. Therefore, the 3-shingles for this document would be:

{"bit", "it-", "t-b", "-by", "by-", "y-b", "-bi"}

One of the issues while making shingles is how to deal with white spaces. A possible solution, which is used commonly, would be to replace all the continuous white space characters with a single blank space.

So, how do you use shingles to find similar documents?

You may convert the documents into shingles and check if the two documents share the same shingles.

An interesting issue here would be to determine the size of the shingle. If the size of the shingle is small then most of the documents would have those shingles, even if they are not identical. For example, if you keep a single size of k=1, then almost all the characters would be shingles and these shingles would be common in most of the documents, as most of them will have almost all the alphabets. On the other hand, bigger shingles may not be able to distinguish similar items. The ideal size of a shingle is k=5 to 9 for small to large documents.

How many different shingles are possible for a character set? Assume that a typical language has n characters and the size of shingles is k, then the possible number of shingles is n^k. Thus, the number of possible shingles in a document may be very large.

Consider that you are using 9-shingles for finding similar research articles amongst very large size articles. The possible character set would require 26 alphabets of the English language and one character for the space character. Therefore, the maximum possible set of shingles would have $(26+1)^9 = 27^9$ 9-shingles with each shingle being 9 bytes long (assuming 1 alphabet = 1 byte). This is a very large set of data.

One of the ways of reducing the data would be to use hash functions instead of shingles. For example, assume that a hash function maps a 9-byte substring to an integral hash bucket number. Assuming that the size of the integer is 4 bytes, then the hash function maps these 27^9 possible 9-shingles to $2^{4*8}-1=2^{32}-1$ possible buckets. You may now use the bucket number as the shingle itself, thus reducing the size of the shingle from 9 bytes to 4 bytes.

For certain applications, such as finding similar news articles, shingles are created based on the words. These shingles are found to be more effective for finding the similarity of news articles.

9.4.2 Minhashing

As you can observe, the set of shingles of a document, in general, is large. Even if we reduce this set using minhashing of shingles to a smaller size, as discussed in the previous section, the size is substantial. In fact, in general, the total size of the set of shingles in Bytes is larger than the size of the document. Thus, it will not be possible to accommodate the shingles in the memory of a computer, so as to find the similarity of the documents. Is it possible to reduce

this stored size of shingles, yet not compromise too much on the estimates of Jaccard similarity?

The method used to do so is called minhashing. The idea here is to convert the set of shingles to a set of signatures of documents using a large number of permutations of rows of the matrix. In order to explain this process, let us use a visual representation of shingles using a matrix. Please note the word visual representation, as it is being used only for explanation. The actual representation would be closer to the representation of a sparse matrix. Consider a universal document that has 5 elements or shingles. The four document sets include the following shingles

Document set 1 = {2, 3, 5}
Document set 2 = {3, 4}
Document set 3 = {1, 4}
Document set 4 = {1, 2, 4}

These documents can be represented using the following matrix of Shingles and the documents:

Shingle/Documents	Set 1	Set 2	Set 3	Set 4
1	0	0	1	1
2	1	0	0	1
3	1	1	0	0
4	0	1	1	1
5	1	0	0	0

Figure 1: Matrix Representation of Documents and associated shingles

You can compute the Jaccard similarity of these documents or sets as follows:

$$\text{JS}_{\text{SetA,SetB}} = \frac{\textit{The number of rows having } 1\textit{'s in columns of BOTH the sets}}{\textit{The number of rows having } 1\textit{' s in at least one of the two column of the sets}}$$

Please note that in the equation given above the columns of the sets having 0s in both columns are not counted, as they show that that shingle is NOT present in both columns.

Thus, for computing the similarity of set 1 and set 2 (see Figure 1), you may observe that row 1 has values 0 in the columns of set 1 and set 2 both, therefore, would not be counted. You may also observe that only row 3 has a value of 1 in columns of set 1 and set 2 both. In all the other rows only one of the two columns is 1. This indicates that only shingle 3 is present in both documents. Thus, the similarity would be:

$$\text{JS}_{\text{Set1,Set2}} = \frac{1}{4};\ \text{JS}_{\text{Set1,Set3}} = \frac{0}{5};\ \text{JS}_{\text{Set1,Set4}} = \frac{1}{5}$$

$$\text{JS}_{\text{Set2,Set3}} = \frac{1}{3};\qquad \text{JS}_{\text{Set2,Set4}} = \frac{1}{4}$$

$$\text{JS}_{\text{Set3,Set4}} = \frac{2}{3}$$

You can verify the Jaccard similarity from the set definitions also. Thus, using the matrix representation, you may be able to compute the similarity of two documents or sets.

After going through the representation and its Jaccard similarity, next, let us define the term minhashing. You can create a large number of orderings of the rows of the shingle/document matrix given in Figure 1, to generate a new sequence of rows or shingles (for our example). The first non-zero shingle of each set is called the minhashed value of that set. For example, consider the following three ordering of the matrix:

ORDER	Shingle/Documents	Set 1	Set 2	Set 3	Set 4
1st	2	1	0	0	1
2nd	4	0	1	1	1
3rd	3	1	1	0	0
4th	5	1	0	0	0
5th	1	0	0	1	1
MH_1(Set n)	-	1st	2nd	2nd	1st

ORDER	Shingle/Documents	Set 1	Set 2	Set 3	Set 4
1st	5	1	0	0	0
2nd	1	0	0	1	1
3rd	3	1	1	0	0
4th	2	1	0	0	1
5th	4	0	1	1	1
MH_2(Set n)	-	1st	3rd	2nd	2nd

ORDER	Shingle/Documents	Set 1	Set 2	Set 3	Set 4
1st	1	0	0	1	1
2nd	2	1	0	0	1
3rd	4	0	1	1	1
4th	3	1	1	0	0
5th	5	1	0	0	0
MH_1(Set n)	-	2nd	3rd	1st	1st

Figure 2: Computation of Minhash Signatures

You may collect these minhashed values, called signatures, into a signature matrix. Thus, a typical signature matrix of the minhasing as above is given in Figure 3.

Set 1	Set 2	Set 3	Set 4
1st	2nd	2nd	1st
1st	3rd	2nd	2nd
2nd	3rd	1st	1st

Figure 3: Minhash Signature Matrix

Likewise, you can create a large number of minhashed values. But how can you compute the Jaccard similarity using these minhashed values?

In order to answer this question, you may think about the probability of finding the same minhash signatures in two columns. This would happen when both the columns have value 1 in the row, which has been selected as the first non-zero row for both columns. Further, the minhash signatures will be different if you have selected a row, which has only one column having a value 1 and the other having a value 0. Thus, going by this logic, if you select a very large number of orderings of the row, the probability of a similar minhash value of two columns would be the same as the Jaccard similarity. Thus, you can reduce the problem of finding the similarity of documents to finding the similarity of minhash signatures. Thus, reducing the complexity of the problem.

For example, you can compute the approximate similarity using Figure 3, as follows:

$$AJS_{Set1,Set2} = \frac{0}{3};\ AJS_{Set1,Set3} = \frac{0}{3};\ AJS_{Set1,Set4} = \frac{1}{3}$$

$$\text{AJS}_{\text{Set2,Set3}} = \frac{1}{3}; \qquad \text{AJS}_{\text{Set2,Set4}} = \frac{0}{3}$$
$$\text{AJS}_{\text{Set3,Set4}} = \frac{2}{3}$$

Thus, you may observe that the Jaccard Similarity can be computed approximately using the signature matrix, which can be used to determine the similarity between two documents.

Now, consider the case when you are computing the signatures for 1 million rows of data. The ordering of these itself is time-consuming and representing an ordering will require large storage space. Thus, for real data, the presented algorithm may not be practical. Therefore, you would like to simulate the ordering using simple hash functions rather than actual ordering. You can decide the number of buckets, say 100, and hash the matrix (see Figure 1) into these buckets. You must select different hash functions such that a column of a row (if it has a value 1) is mapped to a different bucket by a particular hash function. This will ensure that hash functions create different ordering in the buckets. Please remember that hashing may result in collisions, but by using many hash functions, the effect of collision can be minimized. You must select the smallest bucket number to which a column is mapped as its minhash signature value. The following example explains the process with the help of two hash functions. Consider the documents and shingles given in Figure 1, and assume two hash functions given below:

$$h_1(x) = (x + 1)\ mod\ 5$$
$$h_2(x) = (2x + 3)\ mod\ 5$$

We assume 5 buckets, numbered 0 to 4. The following signatures can be created by using these hashed functions:

Initial Values:

	Set 1	Set 2	Set 3	Set 4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Figure 4: Initial hashed values

Now, apply the hash function on the first row of Figure 1. Since Set 1 and Set 2 have values 0, therefore, the value in the above table will not change. However, Set 3 and Set 4 would be mapped as follows:

For Set 3 and Set 4:

$$h_1(1) = (1 + 1)\ mod\ 5 = 2$$
$$h_2(1) = (2 * 1 + 3)\ mod\ 5 = 0$$

Figure 4 will change now to:

	Set 1	Set 2	Set 3	Set 4
h_1	∞	∞	2	2
h_2	∞	∞	0	0

Figure 5: Hashed signature after hashing the first row

Applying the hash function on the second row of Figure 1:

For Set 1 and Set 4:

$$h_1(2) = (2 + 1)\ mod\ 5 = 3$$
$$h_2(2) = (2 * 2 + 3)\ mod\ 5 = 2$$

Since Set 4 already has lower values than these so no change will take place in set 4. Figure 5 will be modified as:

	Set 1	Set 2	Set 3	Set 4
h_1	3	∞	2	2
h_2	2	∞	0	0

Figure 6: Hashed signature after hashing the second row

Applying the hash function on the third row of Figure 1:
For Set 1 and Set 2:

$$h_1(3) = (3 + 1)\ mod\ 5 = 4$$
$$h_2(3) = (2 * 3 + 3)\ mod\ 5 = 4$$

Since Set 1 already has lower values than these so no change will take place in set 1. Figure 6 will be modified as:

	Set 1	Set 2	Set 3	Set 4
h_1	3	4	2	2
h_2	2	4	0	0

Figure 7: Hashed signature after hashing the third row

Applying the hash function on the fourth row of Figure 1:
For Set 2, Set 3 and Set 4:

$$h_1(4) = (4 + 1)\ mod\ 5 = 0$$
$$h_2(4) = (2 * 4 + 3)\ mod\ 5 = 1$$

This will result in changes in the h_1 values of Set 2, Set 3 and Set 4 and h_2 value of Set 2 only, as h_2 values of Set 3 and Set 4 are already lower. Figure 7 will be modified as:

	Set 1	Set 2	Set 3	Set 4
h_1	3	0	0	0
h_2	2	1	0	0

Figure 8: Hashed signature after hashing the fourth row

Applying the hash function on the fifth row of Figure 1:
For Set 1 only:

$$h_1(5) = (5 + 1)\ mod\ 5 = 1$$
$$h_2(5) = (2 * 5 + 3)\ mod\ 5 = 3$$

This will result in changes in the h_1 values of Set 1 only. Figure 8 will be modified as:

	Set 1	Set 2	Set 3	Set 4
h_1	1	0	0	0
h_2	2	1	0	0

Figure 9: Hashed signature after hashing the fifth row

The signature so computed will give almost the same similarity measure as earlier. Thus, we are able to simplify the process of finding the similarity between two documents. However, still one of the problems remains, i.e., there are a very large number of documents between which the similarity is to be checked. The next section explains the process of minimizing the pairs that should be checked for similarities.

9.4.3 Locality Sensitive Hashing

The signature matrix as shown in Figure 9 can also be very large, as there may be many millions of documents or sets, which are to be checked for similarity. In addition, the number of minhash functions that can be used for these documents may be in the hundreds to thousands. Therefore, you would like to compare only those pairs of documents that have some chance of similarity. Other pairs of documents will be considered non-similar, though there may be small false negatives in this group. Locality-sensitive hashing is a technique to find possible pairs of documents that should be checked for similarity. It takes the signature matrix, as shown in Figure 9, as input and produces the list of possible pairs, which should be checked for similarity. The locality-sensitive hashing uses hash functions to do so.

The basic principle of the Locality-sensitive hashing technique is as follows:

Step 1: Divide the overall set of the signature matrix into horizontal portions of sets, let us call them horizontal bands. This will reduce the effective size of data that is to be processed at a time.

Step 2: Use a separate hash function to hash the column of a horizontal band into a hash bucket. It may be noted that if two sets are similar, then there is a very high probability that at least one of their horizontal band will be hashed by some hash function to the same bucket.

Step 3: Perform the similarity checking on the pairs of sets collected in each bucket. You may please note that the probability of hashing dissimilar sets to the same bucket is low.

Thus, finally reducing the pairs that are to be checked for similarity can be reduced using locality-sensitive hashing.

You may please note that this method is an approximate method, as there would be a certain number of false positives as well as false negatives. However, given the size of the data, a small probability of errors are acceptable in the results.

Check Your Progress 2:

Question 1: What is a shingle? What are its uses? List all the 5-shingle of a document, which contain the text “This is a test”

Question 2: Consider the following Matrix representation of 2 documents and their associated shingles. What is the Jaccard Similarity of the documents?

Shingle/Documents	Set 1	Set 2
1	0	0
2	1	1
3	1	1
4	1	1
5	1	0

Question 3: Consider any three orderings of the sets to compute the minhash signature of the documents.

Question 4: Compute the signature matrix from minhash signatures and compute the similarity using the signature matrix.

Question 5: What is Locality sensitive hashing?

9.5 Distance Measures

In the previous sections, we used the Jaccard similarity to find similarity measures. In this section, we define some of the important distance measures that are used in data science. This list is not exhaustive, you may find more such measures from further readings.

The term distance is defined with reference to space, which is defined as a set of points.

A distance measure, say *distance(x, y)*, is the distance between two points in space. Some of the basic characteristics of this distance are:

1. Distance is always positive. It can be zero, only if *x* and *y* are the same points.
2. The distance measured between two points is symmetrical i.e. *distance(x, y)* is the same as *distance(y, x,)*.
3. The distance between two points would be the distance of the shortest path between two points. This can be represented with the help of a third point, say t, by the following equation:

$$distance(x, y) \leq distance(x, t) + distance(t, y)$$

Let us discuss some of the basic distance measures in the following sub-sections.

9.5.1 Euclidean Distance

In Euclidean space, a point can be represented as an n-dimensional vector. For example, a point *x* can be represented as an n-dimensional vector as $x(x_1, x_2, x_3, \ldots x_n)$. The distance of two n-dimensional points *x* and *y* in Euclidean n-dimensional space, where *x* is represented as $x(x_1, x_2, x_3, \ldots x_n)$ and *y* is represented as $y(y_1, y_2, y_3, \ldots y_n)$, can be computed using the following equation:

$$distance(x, y) = \left(\sum_{i=1}^{n} [\{|(y_i - x_i)|\}^m]^{1/m} \right)$$

In the case of m = 1, you get the formula:

$$manhattendistance(x, y) = \left(\sum_{i=1}^{n} |(y_i - x_i)| \right)$$

This is called the Manhattan distance.

In case of m =2, we get the formula:

$$distance(x, y) = \left(\sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \right)$$

You can verify that these measures satisfy all the properties of the distance measure.

9.5.2 Jaccard Distance

As discussed earlier, you compute the Jaccard similarity of two sets, namely set A and set B, using the following formula:

$$\mathrm{JS}_{\mathrm{A,B}} = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard distance between these two sets is computed as:

$$JaccardDistance(A,B) = 1 - \mathrm{JS}_{\mathrm{A,B}} = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Does this distance measure fulfil all the criteria of a distance measure?
The value of Jaccard Similarity varies from 0 to 1, therefore, the value of Jaccard distance will be from 1 to 0. The value 0 of Jaccard distance means that the value of $A \cap B$ will be the same as $A \cup B$, which can occur only if set A and set B are identical. In addition, as set intersection and union both are symmetrical operations, therefore the Jaccard distance is also symmetrical. You can also test the third property using certain values for three sets.

9.5.3 Cosine Distance

The Cosine distance is computed for those spaces where points are represented as the direction of vector components. In such cases, the cosine distance is defined as the angle between the two points. You can use the dot product and magnitude of the vectors to compute the cosine distance between two vectors. For example, consider the two vectors A[1,0,1] and B[0,1,0], the cosine distance between the two vectors can be computed as:

The dot product of the two vectors A.B = 1×0+0×1+1×0=0

$$A.B = |A| \times |B| \cos\theta$$

$$0 = \sqrt{2} \times 1 \cos\theta$$

$$\cos\theta = 0 \; or \; \theta = 90°$$

You may check if this measure also fulfils the criteria of the measures. It may be noted that the range of cosine distance is evaluated between 0 and 180 degrees only. You may go through further readings for more details.

9.5.4 Edit Distance

The edit distance may be used to determine the distance between two strings. It can be defined as follows:
Consider two strings, string A consisting of n characters a_1, a_2,…a_n and a string B of m characters b_1, b_2,…b_m, then edit distance between the two strings is the minimum number operations that are required to make strings identical. These operations are either the addition of a single character or the deletion of a single character.
For example, the edit distance between the strings: A= “abcdef” and B= “acdfg”, would be 3, as you can obtain B from A; by deleting b, deleting e and inserting g at the end in string A.

However, finding these operations may not be easy to code, therefore, edit distance can be computed using the following method:
Step 1: Consider two strings - string A consisting of *n* characters a_1, a_2,…a_n and a string B of *m* characters b_1, b_2,…b_m. Find a sub-sequence, which is the longest and has the same character sequence in the two strings.

Use the deletion of character operation to do so. Assume the size of this sub-sequence is *ls*

Step 2: Compute the edit distance using the formula:

$$editdistance(A, B) = n + m - 2 \times ls$$

For example, in the strings A= "abcdef" and B= "acdfg", the longest common sub-sequence is "acdf", which is 4 characters long, therefore, the edit distance between the two strings is:

$$editdistance(A, B) = 6 + 5 - 2 \times 4 = 3$$

You can verify that edit distance is a proper distance measure.

9.5.5 Hamming Distance

Hamming distance is one of the most used distances in digital design and error detection in digital systems, e.g. while mapping of min-terms into Karnaugh's map or single error correcting code. The Hamming distance is defined as the difference between the components of two vectors, especially when these vectors are of type Boolean.

For example, consider the following two Boolean vectors:

Vector A	1	0	1	1	0	0	1	1
Vector B	1	1	0	1	0	0	0	0
Difference	N	Y	Y	N	N	N	Y	Y

The Hamming distance between the vectors A and B is 4.
You may verify that Hamming distance also satisfies the properties of a distance measure.

You may use any of these distance measures based on the type of data being used or the type of problem.

9.6 INTRODUCTION TO OTHER TECHNIQUES

Big data analytics are the processes that are used to analyse Big data, which include structured and unstructured data, to produce useful information for organisational decision-making. In this section, we introduce some of the techniques that are useful for extracting useful information from Big data. You may refer to the latest research articles from various journals for more details on such techniques.

Textual Analysis:

Textual data is produced by a large number of sources of Big data, e.g. email, blogs, websites, etc. Some of the important types of analytics that you may perform on the textual data are:

1. Structured Information Extraction from the unstructured textual data: The purpose here is to generate information from the data that can be stored for a longer duration and can be reprocessed easily if needed. For example, the Government may find the list of medicines that are being prescribed by doctors in various cities by analysing the prescriptions given by the doctors to different patients. Such analysis would essentially require recognition of

entities, such as doctor, patient, disease, medicine etc. and then relationships among these entities, such as among doctor, disease and medicine.

2. Meaningful summarization of single or multiple documents: The basic objective here is to identify and report the key aspects of a large group of documents, e.g. financial websites may be used to generate data that can be summarised to produce information for stock analysis of various companies. In general, the summarization techniques either extract some important portions of the original documents based on the frequency of words, location of words etc.; or semantic abstraction-based summaries, which use Artificial Intelligence to generate meaningful summaries.
3. Question-Answering: In the present time, these techniques are being used to create automated query-answering systems. Inputs to such systems are the questions asked in the natural languages. These inputs are processed to determine the type of question, keywords or semantics of the questions and possible focus of the question. Next, based on the type of the Question-Answering system, which can be either an Information Retrieval based or a knowledge-based system, either the related information is retrieved or information is generated. Finally, the best possible answer is sent to the person, who has asked the question. Some examples of Question-answering systems are – Siri, Alexa, Google Assistant, Cortona, IBM Watson etc.
4. Sentiment Analysis: Such analysis is used to determine the opinion or perception of feedback about a product or service. Sentiment analysis at the document level determines if the given feedback is positive feedback or negative feedback. However, techniques have been developed to perform sentiment analysis at the sentence level or aspect level.

Audio-Video Analysis:

The purpose of audio or video analysis is to extract useful information from speech data or video data. Most of the calls to the customer call centres are recorded for performance enhancement. One of the common approaches to dealing with such data is to transcribe the speech data using automated speech recognition software. Such software converts the speech into text, which is checked in a dictionary to ascertain if such a word exists or not. Even phonetics can be used for converting speech to text.
Video analysis is performed on video or closed-circuit television (CCTV) data. The objective of such data analysis is to check for security breaches, which may be checked by changes in CCTV data if any. In addition, it can be used to perform indexing of videos, so as to find relevant information at a later time.

Social Media Data Analysis:

This is one of the largest chunks of present-day data. Such data can be part of social networks, micro-blogging, wiki-based content and many other related web applications. The challenge here is to obtain structured information from noisy, user-oriented, unstructured data available in a large number of diverse

and dispersed web pages to produce information like finding a connected group of people, community detection, social influence, link predictions etc. leading to applications like recommender systems.

Predictive Analysis:

The purpose of such analysis is to predict some of the patterns of the future based on the present and past data. In general, predictive analysis uses some statistical techniques like moving averages, regression and machine learning. However, in the case of Big data, as the size of the data is very large and it has low veracity, you may have to develop newer techniques to perform predictive analysis.

Supervised and Unsupervised Learning:

Some of the Big data problems can also be addressed using machine learning algorithms, which create models by learning from the enormous data. These models are then used to make future predictions. In general, these algorithms can be classified into two main categories – Supervised Learning algorithms and unsupervised learning algorithms.

Supervised Learning:
Supervised learning uses already available knowledge to generate models, one of the most common examples of supervised learning is spam detection in which the word patterns and anomalies of already classified emails (spam or not-spam) are used to develop a model, which is used to detect if a newly arrived email is spam or not. Supervised learning problems can further be categorised into classification problems and regression problems.

Unsupervised Learning:
Unsupervised learning generates its own set of classes and models, one of the most common examples of unsupervised learning is creating a new categorisation of customers based on their feedback on different types of products. This new categorisation may be used to market different types of products to these customers. Such types of problems are also called clustering problems.

You can obtain more details on supervised and unsupervised learning in the AI and Machine learning course (MCS224).

Check Your Progress 3:

1. What is the purpose of a distance measure? What are the characteristics of distance measures?
2. What is the Cosine distance measure? How is it different to Hamming's distance?
3. Explain the purpose of text analysis.

9.7 SUMMARY

This unit introduces you to basic techniques for finding similar items. The Unit first explains the methods of finding similarity between two sets. In this context, the concept of Jaccard similarity, document similarity and collaborative similarity are

discussed. This is followed by a detailed discussion on one of the important set similarity applications – document similarity. For finding document similarity of a very large number of documents, a document is first converted to a set of shingles, next the minhashing function is used to compute the signatures of document sets against these shingles. Finally, locality-sensitive hashing is used to find the sets that must be compared to find similar documents. The locality-sensitive hashing greatly reduces the number of documents that should be compared to find similar documents. The Unit then describes several common distance measures that may be used in different types of Big data analysis. Some of the distance measures defined are Euclidean distance, Jaccard distance, Cosine distance, Edit distance and Hamming distance. Further, the unit introduces you to some of the techniques that are used to perform analysis of big data.

9.8 SOLUTIONS/ANSWERS

Check Your Progress 1:

1. For finding similar items, e.g. web pages, the first problem is the representation of web pages into the sets, which can be compared. The major problem, in finding similar items, is the size of the data. For example, if you find duplicate web pages, you may have to compare billions of pairs of web pages.
2. You can define Jaccard's similarity between two sets, viz. Set A and Set B as:

$$JS_{A,B} = \frac{|A \cap B|}{|A \cup B|}$$

Consider two sets A={a, b, c, d, e, f, g} and B={a, c, e, g, i}, then Jaccard similarity between the two sets is:

$$JS_{A,B} = \frac{|\{a,c,e,g\}|}{|\{a,b,c,d,e,f,g,i\}|} = \frac{4}{8} = \frac{1}{2}$$

3. Collaborative filtering is the process of grouping objects based on certain criteria and providing possible suggestions to those objects based on the activities of other objects in the group. The similarity is the basis on which these groups are constructed.

Check Your Progress 2:

1. Shingle is a small string of size 3-9 characters, which can be part of a document. Shingles are used to convert a document into sets, which can be checked for similarity. The document containing the text "This is a test document" has the following set of 5-shingles (_ is used to represent a space character):
   {This_, his_i, is_is, s_is_a, _is_a, is_a_, s_a_t, _a_te, a_tes, _test}
2. Jaccard Similarity of the document is:

$$JS_{Set1,Set2} = \frac{|\{Row2, Row3, Row4\}|}{|\{Row2, Row3, Row4, Row5\}|} = \frac{3}{4}$$

3. The minhash signatures for the following three orderings are shown:

ORDER	Shingle/Documents	Set 1	Set 2
1st	1	0	0
2nd	4	1	1
3rd	3	1	1
4th	5	1	0

5th	2	1	1
MH_1(Set n)	-	2nd	2nd

ORDER	Shingle/Documents	Set 1	Set 2
1st	4	1	1
2nd	1	0	0
3rd	2	1	1
4th	5	1	0
5th	3	1	1
MH_1(Set n)	-	1st	1st

ORDER	Shingle/Documents	Set 1	Set 2
1st	5	1	0
2nd	4	1	1
3rd	3	1	1
4th	2	1	1
5th	1	0	0
MH_1(Set n)	-	1st	2nd

4. The minhash signature matrix is:

Set 1	Set 2
2nd	2nd
1st	1st
1st	2nd

Similarity using signatures = 2/3

5. The locality sensitive hashing first divides the signature matrix into several horizontal bands and hashes each column using a hash function to hash buckets. The similarity is checked only for the documents that hash into the same bucket. It may be noted that the chances of similar document sets may get to the same hashed bucket by at least one hash function is high, whereas the documents that are not similar have low chances of hashing into the same bucket.

Check Your Progress 3:

1. Distance measures are used to find differences between two entities based on some criteria. They are used in problems that measure similarity, which is complementary to distance. The distance measure should be such that it is positive, symmetric and trigonometric (that is the distance between two points is less than or equal to the sum of the distance from the first point to a third point and the distance of the third point to the second point.)
2. Cosine distance measures the angular distance from a specific reference point. It is in general in the range from 0 to 180 degrees. Hamming distance is normally used as the binary distance between two Boolean vectors.
3. Some of the common uses of text analysis are the extraction of structured information from unstructured text, meaningful summarization of documents, question-answering systems and sentiment analysis.

9.9 REFERENCES/FURTHER READINGS

1. Leskovec J., Rajaraman R, Ullman J, **Mining of Massive Datasets,** 3rd Edition, available on the website http://www.mmds.org/
2. Gandomi A., Haider M., **Beyond the hype: Big data concepts, methods, and analytics**, International Journal of Information Management, Volume 35, Issue 2, 2015, Pages 137-144, ISSN 0268-4012.