CHAPTER 7

MODEL BASED LINK DETECTION SYSTEMS

7.1 INTRODUCTION

Story link detection involves the task of comparing and finding out if two given query documents are discussing the same event. The most used and efficient method for link detection is to use query expansion. The basic block diagram of story link detection using query expansion technique, as elaborated in the earlier chapter, has scope for improvement in three blocks. These blocks are: (i) Retrieval System that retrieves the documents relevant to the query documents, (ii) Model Constructor that builds models from the retrieved documents representing the query documents and (iii) Similarity Measure that compares the models. In the previous chapter we have addressed the first issue, where we have proposed methods for improving the retrieval systems. To address the third issue we used MFractional Similarity measure proposed in Chapter 5. The remaining block that requires attention is the model construction block. Information retrieval method is used to obtain documents relevant to each of the query documents. Terms from each of these relevant sets should be assigned proper weights according to their importance to prepare the model corresponding to the query document. The philosophy of these tasks is to ensure that each of the prepared models correctly represents the correct query document. The models contain terms that add additional background information about the event to which the query document belongs. This, doubtlessly, plays a very important role in story link detection systems. We need to apply a proper term weighting scheme for this purpose.
Once the models are ready, these can be compared using MFractional Similarity.

There are various options for assigning weights to the terms to build model from retrieved relevant document set. One of them may be to use only the set of retrieved relevant documents and find the term weights according to various statistics of the terms in the set. Another may be to retrieve documents positive as well as negative to the query document and to assign weights to a term according to its presence in these documents. A possible third approach is to keep the original document and build a model around it. In this chapter we will explore the first two approaches. We propose to use Cohesion model for the first approach and SVM for the second approach.

Section 7.2 explores the use of cohesion modeling technique for model creation. The section shows the effectiveness of this modeling through the experimental results. Section 7.3 shows how SVM can be exploited for building models for story link detection. Named entity has its own place in link detection. We propose to exploit the advantage of named entities using Context Graph, the details of which are elaborated in the subsections. Representation using Context graphs enhances the output of the system using SVM. This sections also presents our proposed approach of combining both SVM and context graph thereby improving link detection system. The last section draws the conclusion.

7.2 COHESION MODEL

The essential goal of research in building a model from relevant documents as used in story link detection is to achieve a good term weighting scheme. This task, in turn, is based on the inter–relationships of the terms
involved in the relevant document set, thereby obtaining a correct representative term weight for each of the terms in the relevant document set. This process bears a very close similarity to our social system, where each person in the society has a representative contribution on an individual basis. Maria Elena Figueroa et al (2002) describes an iterative process of producing social change in a community that improves the health and welfare of all of its members. Among other factors, it discusses social cohesion to be one important measure that can be used to understand the contribution of individuals in the society. Social cohesion is expressed as an important antecedent and consequence of successful collective action.

Social cohesion consists of the forces that act on members of a group or community to remain in, and actively contribute to, the group. We find that the same considerations are applicable to the terms in a document set and map members in the society to the terms, while social cohesion of individuals to a term’s contribution in the documents relevant to the model. Thus the contribution of terms in the relevant document set can be evaluated with respect to the factors discussed in this work. Term weights can be assigned according to different social cohesion factors and the model $M_q$ can be built based on the weights assigned.

Of the six factors mentioned in social cohesion, we find three to strongly influence the process of document expansion as applied to SLD systems. First we discuss these three factors as they have been projected in the social perspective followed by a short description of how these are relevant to the present purpose.

- Sense of belonging – It is the extent to which individual members feel as if they are an important part of the group or community. We find this is equivalent to the contribution of
a term in the document level with respect to the documents in the relevant set. It is a measure of the affinity of a term and hence is called Affinity.

- Feeling of morale – This refers to the extent to which members of a group or community are happy and proud of being a member. This also means that a member who is proud to be associated with a community would belong to only that group. In our case it is equivalent to a term’s presence to only one event and not in many events. It is a measure of the contribution of a term with respect to other events and hence we prefer the name to be Negative Presence.

- Goal consensus – It is the degree to which members of the community agree on the objectives to be achieved by the group. Here we translate it to a measure of how many times each term is repeated in relevant documents, i.e., it is a term’s presence in the relevant document set; the more times a term appears in the relevant document set, the more relevant it is. We call this Positive Presence.

What follows is a description of how these above–mentioned factors can help in capturing a term’s contribution in a model.

**Affinity** – This can be directly mapped with the term’s frequency in the relevant documents.

\[
\text{tf} (t_i) = \frac{1}{n} \sum_{d \in \text{Rel}_n \text{Set}} \text{tf}(t_i, d_i)
\]  
(7.1)
where

\( k = 1..|\text{Tot\_Term}| \)

\( i = 1..n \)

\( n = \) total number of documents in Rel_Set

\(|\text{Tot\_Term}| = \) total number of terms in relevant documents

\( \text{tf (} t_k \text{)} = \) term frequency of \( k \text{th} \) terms in Rel_Set

\( \text{Rel\_Set}_q = \) set of relevant documents for query document \( D_q \)

\( \text{tf}(t_k,d_i) = \) term frequency of \( k \text{th} \) term in document \( d_i \)

**Negative Presence** – Inverse document frequency (idf) perfectly fits this description. Presence of a term in all or most of the documents across the boundaries of groups in the collection can be viewed as lack of willingness and enthusiasm to identify itself with the group.

**Positive Presence** – We calculate the positive presence of a term by the document frequency (df) of the term in the relevant document set using Equation 7.2. More number of times the term is repeated, more overlap is expected among the relevant document set.

\[
df (t_k) = \frac{1}{n} * \text{docfreq}(t_k)
\]  

(7.2)

where

\( \text{df (} t_k \text{)} = \) document frequency of \( k \text{th} \) term

\( \text{docfreq}(t_k) = \) number of documents where term \( t_k \) appears in relevant documents \( \text{Rel\_Set}(D_q) \)

\( n = \) Total number of documents in \( \text{Rel\_Set}(D_q) \)
Each term is given weight as the weighted sum of the tf, idf and df. Adding all the parameters in the calculation of a term’s weight, the final weight of a term is obtained from Equation 7.3.

\[ w(t_k) = c1*tf(t_k)+c2*idf(t_k)+c3*df(t_k) \]  

(7.3)

where

- \( w(t_k) \) – weight of term \( t_k \)
- \( tf(t_k) \) – term frequency of term \( t_k \) in relevant documents
- \( idf(t_k) \) – inverse document frequency of term \( t_k \) in relevant documents
- \( df(t_k) \) – document frequency of term \( t_k \) in relevant documents
- \( c1-c3 \) – constants

Constants \( c1 \) to \( c3 \) are selected empirically.

### 7.2.1 Experimental Results

Input to the Cohesion model is the set of relevant documents retrieved from the corpus. As explained in Chapter 3, we use a corpus truncated on the basis of the deferral period. We call this corpus \( CP_{truncated} \).

For the first set of experimentations, we retrieved relevant documents \(|Rel\_Set|_{D_q}\) from \( CP_{truncated} \) using only MFractional Similarity as shown below.

\[
\text{if} \ (\text{Mfraction}(D_q, d_i) > \text{threshold}), \\
|Rel\_Set|(D_q) = |Rel\_Set|(D_q) \cup d_i;
\]
where $M_{\text{fraction}}(D_q, d_i)$ is the MFractional Similarity value between $D_q$ and $d_i$.

The value for threshold has been empirically determined. The weights of the terms belonging to this set are obtained using cohesion model by applying Equation 7.3.

In Chapter 6, we have proposed two methods of enhancing the retrieval of documents. For the second set of experimentations, we have applied the first of the two methods, viz., IIR and retrieved the relevant documents set and obtained the weights of the terms in this set by applying Equation 7.3. As the next method, we used EBIR and followed the same procedure to find the weights of the terms.

For each of the above methods, we built the models for the two query documents and compared these models using MFractional Similarity.

To apply Equation 7.3 for the calculation of the weights of the terms using Cohesion model, we observe that the equation actually consists of three constants. We provided various weights to the three constants and obtained two systems having different combinations of the factors, viz., $tf+df$ and $tf+idf+df$. We call these linear systems. To compare the performance of these term weighting mechanisms, we have used three traditional existing methods, viz., $tf$, $tf*idf$ and $tf*idf*df$. We call these traditional systems. Table 7.1 shows these five schemes.
Table 7.1 Various Link Detection Systems used for Experiments

<table>
<thead>
<tr>
<th>S.No</th>
<th>SLD Systems</th>
<th>Weighting Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T_tf</td>
<td>tf</td>
</tr>
<tr>
<td>2</td>
<td>T_tfidf</td>
<td>tf*idf</td>
</tr>
<tr>
<td>3</td>
<td>T_tfidfdf</td>
<td>tf<em>idf</em>df</td>
</tr>
<tr>
<td>4</td>
<td>L_tfdf</td>
<td>tf+df</td>
</tr>
<tr>
<td>5</td>
<td>L_tfidfdf</td>
<td>tf+idf+df</td>
</tr>
</tbody>
</table>

The first three systems in the table above are traditional existing systems that do not use the cohesion model. The following two systems are based on Equation 7.3 thereby producing the combinations of social cohesion factors.

First we produce the results of using only MFractional Similarity for retrieval of relevant documents followed by applying each of the methods given in Table 7.1. Table 7.2 shows the break even F1–measure and the corresponding cost and accuracy using the method described. Figures 7.1 to 7.3 show comparison of F1–measure, accuracy, and cost of these systems.

Table 7.2 Performance of Cohesion Model based Link Detection Systems

<table>
<thead>
<tr>
<th></th>
<th>cost</th>
<th>F1–measure</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_tf</td>
<td>0.1295</td>
<td>0.72</td>
<td>0.88</td>
</tr>
<tr>
<td>T_tfidf</td>
<td>0.1333</td>
<td>0.71</td>
<td>0.87</td>
</tr>
<tr>
<td>T_tfidfdf</td>
<td>0.1496</td>
<td>0.70</td>
<td>0.86</td>
</tr>
<tr>
<td>L_tfdf</td>
<td>0.1345</td>
<td>0.74</td>
<td>0.88</td>
</tr>
<tr>
<td>L_tfidfdf</td>
<td>0.1290</td>
<td>0.73</td>
<td>0.88</td>
</tr>
</tbody>
</table>
Figure 7.1 Performance of Link Detection Systems showing F1-Measure

Figure 7.2 Performance of Link Detection Systems showing Accuracy
Comparison Between Various Link Detection Systems

Performance with respect to the F1–Measure of linear tf–df is the best as is evident from Table 7.2 and Figures 7.1 to 7.3. The next best F1–measure is achieved by linear tf–idf–df (L_tfidfdf). Our observation is that the inclusion of idf factor in L_tfidfdf reduces the false positives as well as the true positives. The reduction of the true positives in turn reduces the F1–measure. However the increase in the true negatives (i.e. reduction of the false positives) helps this system to maintain a high accuracy. Less false positives of the linear tf–idf–df (L_tfidfdf) help it to reduce the cost with respect to the linear tf–df.

Traditional tf (T_tf) in its simple form achieves less true positives, but manages to achieve less false positives as well. Hence this system has an increase in the true negatives. This leads to better accuracy value. However due to low true positive values, the system’s F1–measure has reduced.
In link detection system low cost indicates better performance. Systems in link detection are expected to produce less false positives while maintaining high true positives. Hence cost function is designed in favor of less false positives. From Figure 7.3 we observe that the lowest cost is achieved by linear tf–idf–df (L_tfidfidf). Other systems like T_tf, T_tfidf and linear tf–df (L_tfidf) have achieved a comparable performance.

Next we have performed experimentations using IIR and EBIR as the retrieval method followed by implementation of various methods of Table 7.1. However, since the results of the two IR methods, viz., IIR and EBIR, are quite close to each other, and we have a space constraint, we have provided in the dissertation only the results of EBIR and the cohesion model. The next part of this subsection shows the experimental results for cohesion model using EBIR. As is expected from the earlier results, in general, the linear methods have performed better here than the nonlinear ones.

Table 7.3 shows the F1–measure, accuracy and cost measure of these systems. Figures 7.4 to 7.6 show comparison of F1–measure, accuracy and cost measures respectively.

Table 7.3  Performance of Cohesion Model based Link Detection System with EBIR

<table>
<thead>
<tr>
<th>Systems</th>
<th>Cost</th>
<th>F1–measure</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_tf</td>
<td>0.137173</td>
<td>0.72751</td>
<td>0.87364</td>
</tr>
<tr>
<td>T_tfidf</td>
<td>0.141721</td>
<td>0.724936</td>
<td>0.870617</td>
</tr>
<tr>
<td>T_tfidfidf</td>
<td>0.124241</td>
<td>0.737838</td>
<td>0.882709</td>
</tr>
<tr>
<td>L_tfidf</td>
<td>0.116747</td>
<td>0.751696</td>
<td>0.889359</td>
</tr>
<tr>
<td>L_tfidfidf</td>
<td>0.136056</td>
<td>0.73385</td>
<td>0.875453</td>
</tr>
</tbody>
</table>
Comparison Between Various Link Detection Systems

Figure 7.4  F1–Measure of Cohesion Model based Link Detection System with EBIR

Comparison Between Various Link Detection Systems

Figure 7.5  Accuracy of Cohesion Model based Link Detection System with EBIR
Comparison Between Various Link Detection Systems

Figure 7.6  Cost of Cohesion Model based Link Detection System with EBIR

Linear tf–df (L_tfidf) shows the best performance compared to all the other systems as is evident from Figures 7.4 to 7.6. L_tfidf shows an increase in F1–measure at the same cost as the T_tf and T_tfidf, although the accuracy is not comparable.

As an extension of the discussion of the last chapter, we have compared the performance of cohesion model based systems with and without IR. Again, fulfilling our expectations, cohesion model based link detection systems perform much better than systems without any IR. Below we provide comparative evaluations of cohesion model with and without EBIR.
Comparison of Cohesion Model with and without EBIR

Figure 7.7 F1–Measure Comparisons between Model With and without EBIR

Comparison of Cohesion Model with and without EBIR

Figure 7.8 Accuracy Comparisons between Model with and without EBIR
As is evident from the above three Figures 7.7 to 7.9, that using a combination of enhanced IR with cohesion model produces excellent result.

The next section discusses the retrieval of positive as well as negative documents and the use of SVM to detect links between two query documents.

7.3 SVM BASED LINK DETECTION SYSTEMS

Support Vector Machine (SVM) is a popular learning technique for data classification. For its great accuracy and tolerance towards sparse data, it has been used in many classification applications successfully. The goal of SVM is to find a decision surface to separate the training data samples into two classes (positive and negative) and make decisions based on the support vectors that are selected as the only effective elements from the training set. It finds the hyperplane that maximizes the margin of separation. Researchers
have successfully used SVM in text classification as well. For text classification, SVM makes decision based on the globally optimized separating hyperplane. It simply finds out on which side of the hyperplane the test document belongs (Joachims 1997a; 1998). Thus, SVM is a tool that can be used to classify a new document if it is trained on the category information a-priori. Success of SVM in various classification problems induced us to exploit it in the story link detection task. In order to use SVM in story link detection, SVM needs to be trained for the query document, say \( D_{q1} \). Once SVM is trained, then the second query document, \( D_{q2} \) may be treated as the new incoming document and the trained SVM can be used to classify this incoming document to find whether it is relevant to \( D_{q1} \). However, using SVM in this manner in SLD has a difficulty. The SVM cannot be trained on a single query document \( D_{q1} \). For training, it needs pre–labeled positive and negative documents for the class for which it is to be trained, in our case query document \( D_{q1} \). Moreover, in link detection, only unlabelled corpora are used; information even about the labels of the given two documents is unknown. To overcome the difficulty of the unavailability of pre–labeled training data for the corpora of link detection systems, we propose to employ information retrieval system to obtain a relevant set for query document \( D_{q1} \) as is used quite successfully in query expansion technique in link detection system. Good performance of the query expansion technique and the good performance of SVM encouraged us to use SVM for link detection.

There are essentially two differences between what happens in the usual query expansion technique in link detection and what we have employed in our work. The idea of retrieving documents is already present in the query expansion technique; however, we propose to use this method to obtain, instead, positive as well as negative documents for one of the query documents, say \( D_{q1} \). These two sets will act as the training set to train the SVM for this document that is taken as a class. The second difference
between traditional query expansion methods in SLD and our proposed method is that instead of using an explicit model, we exploit the trained SVM to extract a decision as to whether other document $D_{q2}$ can be classified to the class of $D_{q1}$.

The corpus from which positive and negative documents are to be selected is the truncated corpus $C_{P_{\text{truncated}}}$. Our goal is to use this corpus to collect documents positive and negative to $D_{q1}$. $D_{q1}$ is represented using tf. Considering document $D_{q1}$ as query, MFractional Similarity method in combination with a suitable empirically decided threshold is used to obtain documents that are positive to $D_{q1}$ from $C_{P_{\text{truncated}}}$. For example, if Tsumani is a topic, and Tsunami in Chennai in 2005 is an event, then a document that discusses the Tsunami that hit Chennai shores in 2005 and its problems is a positive document.

Finding negative documents requires special attention. Let us consider what set is truly negative as far as training the SVM is concerned. We observe that documents that are completely unrelated to the topic to which $D_{q1}$ belongs are not the correct set of negative documents since they talk of topics that bear no similarity with the topic in hand. We are interested in collecting those documents as negative that are discussing about the same topic but not the same event. Considering the same example of Tsumani as a topic, and Tsunami in Chennai in 2005 as an event, all stories about incidents of Tsumani in other places and/or in other times should be the negative documents. The calamity caused by the earthquake in Maldives is a different topic and is not to be considered as a proper negative document. Therefore, in $C_{P_{\text{truncated}}}$ corpus, documents that are completely unrelated to $D_{q1}$ are not the correct set of negative documents. Thus, when we say negative documents, we expect that there should be some trace of similarity between the event and the document, albeit very less.
This observation has led us to filter the documents that are obtained from the collection within deferral period using similarity measure method.

To implement the filtering of the documents properly and obtaining effective sets of positive and negative sets, we use three thresholds, viz., \( \theta_1, \theta_2 \) and \( \theta_3 \). The method of using the MFractional Similarity technique and the three thresholds is elaborated in the following algorithm.

**Algorithm Find_Rel_NRel_Docs** *(Input: Retrieved documents with similarity measures, Output: Relevant set of documents and non-relevant set of documents)*

Begin

for \( (d_i \in \text{CP}_{\text{truncated}}) \)

if \((\text{Mfraction}(D_{q1}, d_i) > \theta_1)\)

\(|\text{Rel}_\_\text{Set}(D_{q1})| = |\text{Rel}_\_\text{Set}(D_{q1})| \cup d_i;\)

else if \((\text{Mfraction}(D_{q1}, d_i) < \theta_2 \&\& \text{Mfraction}(D_{q1}, d_i) > \theta_3)\)

\(|\text{Neg}_\_\text{Set}_{D_{q1}}| = |\text{Neg}_\_\text{Set}_{D_{q1}}| \cup d_i;\)

else discard \(d_i\).

End

where \(\text{Mfraction}(D_{q1}, d_i)\) is the MFractional Similarity value between \(D_{q1}\) and \(d_i\), \(|\text{Rel}_\_\text{Set}_{D_{q1}}|\) and \(|\text{Neg}_\_\text{Set}_{D_{q1}}|\) are the retrieved positive and the negative document sets respectively.

This process is explained in Figure 7.10.

Figure 7.10 shows documents that have similarity value greater than the threshold \( \theta_1 \) are considered as relevant and positive. These belong to the region P in the figure. From these relevant documents n top documents are chosen as positive documents for training the SVM. Similarly for negative
documents, bottom m documents whose similarity values are less than the threshold $\theta_2$ and greater than the threshold $\theta_3$ are taken as negative document set. These belong to the region N. These sets are used to train the SVM.

![Diagram](image)

**Figure 7.10 Corpus CP\text{truncated} with Documents Relevant and Non–Relevant to D_{q1}

It is to be noted that the region X is the negative documents without the minimum similarity value and hence are discarded from consideration. Similarly, the region A has documents that may either be positive or negative and hence discarded. Thus P forms the positive documents and N forms the negative documents for SVM training.

In our experiments, for training, we have set the n to be 10, since in most of the cases number of positive documents retrieved is less than 10. The value for m is also set to 10 to have more or less equal number of positive and negative documents.

We have used a linear SVM classifier – libsvm 2.83 (Fan et al 2005) for our experiments. Each document is represented using tf*idf in the
vector form and given for training the SVM. After SVM is trained, it acts as a classifier for the document $D_{q1}$. Document $D_{q2}$ is given to the SVM to test whether it can be classified to $D_{q1}$; in other words to check if $D_{q2}$ is similar to $D_{q1}$.

Although SVM is quite successful in generic classification, and there is an obvious closeness of classification and link detection, when we performed our experiments with the SVM training and testing described above, the results were not as good as expected. From this we conclude that SVM as used is unable to take care of link detection technique. We consider two possible factors for this. The first may be that only MFractional Similarity is used to retrieve documents from which the positive and negative documents are obtained for training SVM. An improved retrieval method may improve the overall performance of the tool. The other factor may be due to the lack of opportunity to include the important aspect of including the contributions of named entities in these systems. As we have already established in the earlier chapter, events often take place around some named entities. Thus, although nothing can be decided solely based on named entities, some importance to named entities definitely helps in obtaining a better result in story link detection. While we do not expect that all documents belonging to an event need necessarily contain named entities related to the event, most of the times a story related to an event is expected to answer question of the form *where, when, who*, that come from the named entities. In addition, such documents are also expected to answer questions of the type *what, how*, which are the descriptions of the event and does not involve named entities. Therefore, many event–related documents are expected to be a combination of named entities as well as non–named entities. This information unfortunately cannot be exploited in SVM. We propose to enhance the performance of SVM based link detection systems by employing
an improved IR and also by employing a technique to include the importance of named entities and their co–occurring terms in the given context.

For the first enhancement, we used Entity based IR as described in the earlier chapter. The experimental result sub–section shows the excellent performance of SVM using EBIR. For the second enhancement, we propose to use a graph based representation of the document where named entities are the key elements and terms co–occurring with the named entities are taken as the context. We call this graph thus formed as Context Graph. The next subsection describes the creation of the Context Graph. Combined SVM and Context graph is discussed in the subsection after the next. Experimental results subsection provides the results and also provides a comparative study of the various SVM based methods.

7.3.1 Context Based Graph Representation

To exploit the role of named entity in the description of events and to utilize the dependence of other terms with these named entities, we represent the query documents using context graph. The main idea for using the context graph is to reduce false positives. Negative documents either share some of the keywords or some of the named entities but may not be both. We propose to use context graph based representation to capture this.

We extract named entities and the sentence–level context in which they appear from the query document. Context graph is built around the named entities as the base. The seed for the graph representation is the named entities. We select the other terms one by one.

Figure 7.11 below shows the graph representation of a document.
In the above figure, the shaded nodes indicate the named entities while the others indicate the non–named entities. An edge between the nodes indicates sentence level co–occurrence.

Given a document $D_q$, it is first split into sentences $S_i$. Hence $D_q = \{S_i\}$. Query document $D_q$ may contain two sets of terms, the set of named entity terms called $\text{NE}$ and the set of non–named entity terms called $\text{NNE}$. A sentence $S_i \in D_q$ consists of either only $\text{NNE}$ or a combination of $\text{NE}$ and $\text{NNE}$. The first types of sentences are denoted by $S_{\text{NNE}}$ while the second types of sentences are denoted by $S_{\text{NE}}$. Thus a query document $D_q = \{S_{\text{NNE}} \cup S_{\text{NE}}\}$. We further break these sentences into named entities and non–named entities. While $S_{\text{NNE}}$ consists of only $\text{NNE}$ types of terms, $S_{\text{NE}}$ consists of both the types of terms. Followings are definitions of some basic elements in the document:

Let $S_{\text{NEi}} = \{ \text{NE}_{i1} \cup \text{NNE}_{i2} \}$
if \( x = \{ \text{NE}_{i1} \} \) and \( y_1 = \{ \text{NNE}_{i2} \} \)

then \( S_{\text{NEi}} = \{ x \cup y_1 \} \)

Let \( S_{\text{NNEj}} = \{ \text{NNE}_{j1} \cup \text{NNE}_{j2} \} \) such that \( \{ \text{NNE}_{j1} \} \subseteq \{ \text{NNE}_{i2} \} \)

If \( y_2 = \{ \text{NNE}_{l1} \} \) and \( z = \{ \text{NNE}_{l2} \} \) where \( \{ y_2 \} \subseteq \{ y_1 \} \)

then \( S_{\text{NNEj}} = \{ y_2 \cup z \} \)

Initial graph \( G (V_{\text{initial}}, E_{\text{initial}}) \) is constructed with nodes \( V_{\text{initial}} \) and edges \( E_{\text{initial}} \) as follows:

If \( x = \{ \text{NE}_i \} \) and \( y_1 = \{ \text{NNE}_i \} \) for \( i^{th} \) NE sentence in query document \( D_q \) then,

\[
V_{\text{initial}} = \{ x \cup y_1 \} \forall y_1 \in y_1 \in S_{\text{NEi}}
\]

\[
E_{\text{initial}} = \{ (x, y_1_j) \cup (x, y_1_k) \mid y_1_j, y_1_k \in y_1 \text{ and } j \neq k \}
\]

\[
\forall (x, y_1) \in S_{\text{NEi}} \quad \forall i \text{ such that } S_{\text{NEi}} \in D_q
\]

The second level of the graph is built from the non–named entity term (called \( z \)) that co–occurs with any term common in \( \{ y_1 \in S_{\text{NEi}} \} \) in the \( S_{\text{NNEj}} \) sentences and this set of non–named entity terms is called \( \{ y_2 \} \). Thus the overall graph is described below.

\[
V = \{ V_{\text{initial}} \cup z \}
\]

\[
E = \{ E_{\text{initial}} \cup (y_2, z) \}
\]

\[
\mid y_2, z \in S_{\text{NNEj}}, \forall (y_2, z) \ni (x, y_1) \in E_{\text{initial}}
\]

and \( \{ y_2 \} \subseteq \{ y_1 \} \)

Elements \( x, y, z \) form the vertices \( \{ V \} \). First step for building the graph is to get the named entities \( x \) in the document \( D_q \), which forms first set of nodes for graph \( G \). Then the context of \( x \) is picked up. This is the set of non–entity terms \( y \) that co–occurred with \( x \) in sentence \( S_i \). These are added to
the graph. If we construct the graph just by selecting the named entity and the other terms that co–occurred with named–entity in the sentence level, we may miss out many important descriptions on the event. So we extend the graph to include other sentences, which do not have named entity, but are information bearing sentences. To select those sentences and to expand the context to one more level, we have taken the non–entity terms \( z \) that has co–occurred with \( y \) in the document at the sentence–level. As already established, \( y \) is the set that has co–occurred with \( x \) in some sentence. Thus the document \( D_q \) is represented as graph \( G \) created as above.

Query documents \( D_{q1} \) and \( D_{q2} \) are represented in graph forms \( G_{q1} \) and \( G_{q2} \) and then are compared using the number of overlapping edges as given by Equation 7.4.

\[
S(G_{q1} \text{ and } G_{q2}) = \frac{n(E_{q1} \cap E_{q2})}{\min (n(E_{q1}), n(E_{q2}))} \quad (7.4)
\]

where

\( E_{q1}, E_{q2} \) are the edges in the graph \( G_{q1} \) and \( G_{q2} \) respectively.

Context graphs described above help in recognizing the closeness of the query documents with respect to co–occurring terms related to the named entities of the these documents. Although SVM successfully captures the unigram–related closeness of two documents, context graph measures the closeness with respect to the co–occurring terms.

Weighted sum of the outputs of SVM and Context Graph similarity measure therefore provides necessary information about link between the two query documents. This is used to take the final decision whether the given documents \( D_{q1} \) and \( D_{q2} \) are linked.
7.3.2 Combining SVM and Context Graph

SVM uses the expanded terms from the documents relevant to $D_q$. It not only captures the contribution of a term with respect to an event, it is also capable of correctly manipulating the level of contribution of term, since it compares both positive and negative documents pertaining to an event. However, it fails to capture the importance of terms in the context of named entities present, which is successfully captured by the context graph. Each of the methods individually contributes towards an improved performance of link detection system in different directions. In the combined approach, we calculate the similarity measure for both and use weighted sums of the results. Final similarity measure between the query documents is calculated by Equation 7.5.

$$S(D_{q1}, D_{q2}) = \alpha SO + (1-\alpha) S(G_{q1}, G_{q2})$$  \hspace{1cm} (7.5)

where

- $S(D_{q1}, D_{q2})$ – similarity between input documents $D_{q1}$ and $D_{q2}$
- $SO$ – SVM output
- $S(G_{q1}, G_{q2})$ – graph similarity
- $\alpha$ – control parameter.

Control parameter $\alpha$ in Equation 7.5 decides the weight given to the output of SVM and context–graph. Final link decision is taken if the similarity value, $S(D_{q1}, D_{q2})$ is greater than a predefined threshold.

In the following subsection, we evaluate performances of various SVM based link detection Systems, as described above, on the link detection task.
7.3.3 Experimental Results of SVM with Context–Based Graph

The corpus is the truncated corpus CP_{truncated}. We have performed experiments that implement SVM as described in Section 7.3, the context graph as described in Section 7.3.1 and a combined SVM and Context graph based approach as described in Section 7.3.2.

Table 7.4 Performances of SVM and Graph Based Link Detection Systems

<table>
<thead>
<tr>
<th>Methods</th>
<th>F1–measure</th>
<th>Accuracy</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph–svm</td>
<td>0.7549669</td>
<td>0.8881499</td>
<td>0.1201768</td>
</tr>
<tr>
<td>svm</td>
<td>0.74406</td>
<td>0.88271</td>
<td>0.12637</td>
</tr>
<tr>
<td>graph</td>
<td>0.720109</td>
<td>0.87545</td>
<td>0.13155</td>
</tr>
</tbody>
</table>

Figure 7.12 F1–Measure of the SVM and Graph based Link Detection Systems
Figure 7.13  Accuracy of the SVM and Graph based Link Detection Systems

Figure 7.14  Cost Measure of SVM and Graph based Link Detection Systems
Table 7.4 and Figures 7.12 to 7.14 show the F1–measure, accuracy and cost of link detection system that shows independent as well as combined output of SVM and context graphs. From the experimental results we observe that the combined effect produces better result than the individual efforts of both SVM and context graph. SVM is able to produce very good true positives, i.e., it is able to identify linked documents properly. Moreover it is capable of reducing false positive (false alarm) and false negative (miss) when compared to the other systems. As mentioned above, capturing the named entities and context terms helps to reduce the false positives much. Context graph alone, when used for link detection system, is unable to identify many true positives. The reason behind this is, we miss out many terms in the document. When the context graph is constructed with named entity and the co–occurring terms, and the second level of co–occurring terms, many other sentences, which do not contain first level terms and second level terms are not added in the graph. By this we may miss out some contributing terms, which leads to reduction in true positives. However combining the SVM output and Context Graph output we are able to get impressive results.

Equation 7.5 combines the output of SVM and context graph. Weight given to each output is decided by the control parameter $\alpha$. Table 7.5 shows the performance of the system for different values of alpha.

<table>
<thead>
<tr>
<th>$\alpha$ values</th>
<th>F1–measure</th>
<th>Accuracy</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.743386</td>
<td>0.882709</td>
<td>0.12614</td>
</tr>
<tr>
<td>0.5</td>
<td>0.749669</td>
<td>0.885732</td>
<td>0.12277</td>
</tr>
<tr>
<td>0.25</td>
<td>0.754967</td>
<td>0.88815</td>
<td>0.12018</td>
</tr>
</tbody>
</table>
It can be observed that when $\alpha$ is low, the performance of the system is high. This indicates that when the context graph output is given more weightage, system performs better. However, context graph on its own doesn’t perform well.

Next we provide the results of SVM in link detection using EBIR for retrieval.

7.3.4 Entity based IR with SVM for Link Detection System

In this sub-section, we have compared our results of SVM with and without EBIR.

Table 7.6 provides the list of the systems considered for evaluation.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SVM with Entity–Based IR</td>
</tr>
<tr>
<td>2</td>
<td>SVM without Entity–Based IR</td>
</tr>
</tbody>
</table>

Table 7.7 Performance of Various SVM based Link Detection System with and without EBIR

<table>
<thead>
<tr>
<th>Systems</th>
<th>F1–measure</th>
<th>Accuracy</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>EB–IR_svm</td>
<td>0.757412</td>
<td>0.891173</td>
<td>0.115392</td>
</tr>
<tr>
<td>svm</td>
<td>0.744063</td>
<td>0.882709</td>
<td>0.126373</td>
</tr>
</tbody>
</table>
Comparison of Link Detection System with and without EB-IR

Figure 7.15 F1–Measure of SVM based Link Detection System with and without EBIR

Figure 7.16 Accuracy of SVM based Link Detection System with and without EBIR
Comparison of Link Detection System with and without EB-IR

Table 7.7 and Figures 7.15, 7.16 and 7.17 show the performance of the link detection system using SVM based on Entity Based retrieval system and also provide a comparative study of SVM with and without EBIR. From the table and charts it is clear that the use of EBIR in SVM definitely improves the overall performance. Comparing SVM with EBIR and without EBIR, SVM with EBIR exhibits a better performance than SVM without EBIR. Entity Based information retrieval is able to reduce the noisy documents in the information retrieval stage. This, in turn, helps a better training of SVM and hence the performance improvement.

Now, we show the comparative performances of all the link detection systems that have been proposed in this chapter.
Figure 7.18 F1–Measure Comparison of all Proposed Link Detection Systems

Figure 7.19 Accuracy Comparison of all Proposed Link Detection Systems
Comparison Between Various Link Detection Systems

![Comparison Between Various Link Detection Systems](image)

**Figure 7.20 Cost Comparison Between all Proposed Link Detection Systems**

Figures 7.18 to 7.20 show comparison of F1-measure, accuracy and cost between all the proposed link detection systems respectively. As it is obvious from the figures, SVM with EBIR performance is the best among the proposed link detection systems. As quoted already, SVM is able to identify links properly with better IR provided by EBIR. However, Cohesion Model is not able to produce true positives as good as SVM. Hence its performance is less than SVM.

### 7.4 CONCLUSION

In this chapter we have proposed link detection system called cohesion model that uses the concept of social cohesion. With factors of social cohesion applied to terms of the relevant document we are able to identify important terms and assign weight according to its importance. This helps in achieving better model building and hence a better link detection
system. Results are much better than the base system that used only cosine similarity and also the system with query expansion without cohesion model.

Further in this chapter we have proposed link detection systems that use SVM and context graph for link detection system. With the inspiration of success of SVM in various categorization tasks, we have used SVM–Context Graph based method for story link detection system and as expected obtained good results. SVM is able to produce high true positives. However the performance of the link detection system depends on the quality of the retrieved documents and hence it comes as no surprise that SVM with EBIR performs so well.

Context Graph approach checks for the overlapping edges (bi-gram) rather than overlapping terms. It improves performance by reducing the false positives.

Thus, this chapter provides the culmination of all the research we have performed. The next chapter draws conclusion about the dissertation.