Extractive Summarization From Document Sentence Graphs Using The Influence Maximization Approach

Varun Gangal and Pallavi Gudipati

Department of Computer Science and Engineering
Indian Institute of Technology, Madras
{vgtomahawk,pallavigudipati}@gmail.com

Abstract. In this paper, we apply two techniques prevalent in Social Network Analysis, Community Detection and Influence Maximization, to solve the problem of Extractive Text Summarization. In our approach, we first apply community detection techniques to the sentence graph, followed by PageRank in one technique and Influence Maximization in the other. We also analyze our approaches on TIPSTER's cmp-lg dataset using ROUGE-1, with Global PageRank, Global Influence Maximization and Centroid-based method as baselines. We observe that PageRank with Community Detection works best for undirected and backward-directed graphs, whereas Influence Maximization with Community Detection works best for forward-directed graphs. The experiments demonstrate the advantages of incorporating Community Detection and Influence Maximization into graph-based methods for summarization, to model the information flow and latent concept structure in the document.

Keywords: Text Summarization, Community Detection, Influence Maximization, PageRank, ROUGE

1 Introduction

1.1 What is a Summary?- Principles and Intuition

Though we use the words "summary" or "summarize" often in everyday conversation or writing, it is necessary to define a bit more precisely what we mean by a summary. [12] characterizes a summary as "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that".

It also says that the main goal of a summary is to present the main ideas in a document in less space. In the hypothetical case where all sentences in a document have equal importance, producing a summary would lead to a proportional decrease in informativeness, and would not be of use. However, they observe that **"information content occurs in bursts"** in practice, and one can identify more informative segments of text, and ignore the others when summarizing. The

choice of what these informative segments could be - words, sentences or lexical chains is a **choice of granularity** which we must be aware of when we propose an approach. There are many other distinctions in the nature of summaries - indicative vs informative, topic-oriented(user requirement) vs generic(reflecting the author's point of view). The survey paper [9], lists three important aspects of summarization

- Summaries may be produced from a single document or multiple documents
- Summaries should preserve important information
- Summaries should be short

We need to keep these principles in mind while proposing any new approach.

1.2 Single Document vs. Multi-Document Summarization

Single document summarization is the task of producing a summary from a single document, whereas multiple document summarization involves producing one from a corpus of documents. There are three additional challenges which one must deal with when one does multiple document summarization

- Coping with **redundancy**
- Identifying imporant **differences** between documents
- Ensuring summary **coherence**, even though the documents may come from different sources

Multi-Document Summarization can be done in a **layered** manner, by defining the summary of multiple documents as a **metasummary**. A metasummary is a summary of summaries. This approach treats the problem as first producing single document summaries, and then producing a summary of these summaries. This approach is first proposed in [7]

1.3 Extractive vs. Abstractive Summarization

Extractive summarization creates a summary from a document(or a set of documents) by selecting some sentences from the document to appear in the final summary. Abstractive summarization attempts to produce a summary with completely new sentences that conveys the meaning of the original document. It also puts a strong emphasis on form, aiming to produce a grammatical summary, which usually requires advanced language generation techniques. [2] notes that abstractive summarization requires **"heavy machinery for language generation"**. It also requires knowledge specific to the domain at hand.

One approach to performing abstractive summarization is to first create an extractive summary, and then use then use some abstractive summarization technique. This approach can also possibly be used to **extrinsically evaluate** extractive summarization approaches

2 Literature Survey

2.1 LexRank [1]

In this technique, the centrality of each sentence is assessed and represented as a tf-idf vector. The paper uses a centroid-based summarization as its baseline. This baseline was used in the first web-based multidoc summarization system. The basic premise is that sentences that are more similar to other sentences are more central to the topic. Some of key issues adressed are that of defining similarity and centrality. In degree-based centrality, greater degree nodes are important. The problem with this definition is that it does not take global information into account. It is also very sensitive to the choice of threshold. To get the centrality of the nodes, *LexRank* performs PageRank on the unweighted similarity graph. *Continuous LexRank* improves upon it by considering a weighted similarity graph. LexRank accounts for information subsumption amongst sentences, prevents effect of unnaturally high idf scores and is more robust to noise, when compared to centroid-base method.

2.2 TextRank: Bringing Order into Texts [8]

TextRank is largely similar to LexRank, but uses a slightly more general setting. This paper proposes a graph based ranking model for graphs extracted from natural language texts. It paper also demonstrates the model by using it for the tasks of keyword and sentence extraction. TextRank defines similarity between two sentences based on content overlap. To prevent promotion of long sentences, a normalization factor is used.

$$Sim(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log|S_i| + log|S_j|} \tag{1}$$

This similarity measure is slightly different from the one used in LexRank. LexRank's focus was primarily on the multi-document summarization task, unlike TextRank. LexRank also applied a heuristic post-processing step that builds up a summary by adding sentences in rank order, but discards any sentences that are too similar to ones already placed in the summary.

2.3 ClusterRank: A Graph Based Method for Meeting Summarization [2]

ClusterRank's aim was to summarize transcripts of meetings or conversations, which consist of sequences of utterances, instead of sentences in traditional settings such as news articles. Some of challenges faced in this setting in addition to the traditional challenges is that the sentences are ill formed. Also, individual sentences carry lesser information and has more redundancy due to repetition. The authors construct a graph where every node is a cluster of utterances, rather than an individual utterance. This prevents off-topic utterances from getting

promoted. It also overcomes the problem of lesser information in individual utterances. The paper uses text segmentation to segment the text such that each section has utterances about the same subject. Then a graph is constructed by taking cosine similarity based on bag of words between clusters. The clusters are scored by using PageRank on the cluster graph. Sentences are scored by by taking similarity to cluster centroid. The final summary is created using greedy selection.

2.4 Language Independent Algorithm [7]

This paper proposes an approach that is both unsupervised and language independent. Experiments are done on English and Portuguese datasets. It also deals with both single and multiple document summarization. First, PageRank and HITS are adapted to include edge weights.

$$PR(V_i) = (1-d) + d \sum_{V_j \in In(V_j)} w_{ji} \frac{PR(V_j}{\sum_{V_k \in Out(V_j)} w_{kj}} \tag{2}$$

Similarly, HITS is also modified. For single document summarization, each sentence is represented by a node. After running the ranking algorithm, sentences with highest scores are selected for the summary. Edge weight of (u, v) is the similarity between u and v. The paper uses cosine similarity as the similarity measure. For multiple document summarization, the paper introduces a maximum threshold on the similarity score to avoid redundancy, which is empirically set. No edge is added between the sentences that exceed this threshold.

2.5 Shortest-path Algorithm [13]

Extraction-based summaries are somewhat unappealing to read as there is a lack of flow in the text. The approach proposed by this paper is based on finding the shortest path from the first sentence to the last sentence in the sentence graph. The cost of an edge is based on how similar the sentences are, how far apart they are in the original text, and how relevant the destination node's sentence is to the document. N shortest paths are chosen using a greedy approach and from them, the path closest to the desired length is chosen.

3 Methodology

We follow a four stage approach to perform extractive single document summarization.

- **Sentence Graph Construction**
  Construct a sentence similarity graph, based on an adequate notion of similarity.
- **Community Detection**
  Discover communities in the graph, using a community detection algorithm

- **Budget Allocation**
  Decide budgets, in other words, the number of sentences to be selected for the summary, for each community.
- **Influence Maximization / PageRank**
  Perform influence maximization or run PageRank on each community to find a number of influential nodes which is equal to its budget. Stitch the nodes corresponding to these sentences together to form a summary.

We explain each of these stages in greater detail below.

3.1 Sentence Graph Construction

Sentence graph construction involves significant pre-processing in order to get the words in their correct form, before one can compute the pairwise similarities. The pre-processing required is as follows

Tokenization Tokenization is the process by which we break a document into a sequence of sentences, and subsequently break each sentence into a sequence of words. Tokenization itself involves some important subtasks, such as Sentence Boundary Recognition. For instance, consider the following sentences

H.M Patel represented India at the United Nations.

Here, the first period refers to the initials, whereas the last one actually marks the sentence boundary.

Stemming Stemming ensures that words are mapped to their morphological roots(albeit in a very approximate fashion). Performing stemming is essential since we need to remove variations in tense and voice when computing similarity between sentences based on their word forms. For example, consider the sentences below

- Gita sings a Hindustani classical song every Sunday morning.
- She has been trained in singing from a very young age.

Here, in the absence of stemming(which would reduce singing to sing), we would miss out on the similarity in the verb in these two sentences. We prefer stemming over lemmatization for form-based similarity since lemmatization is an order of magnitude slower in terms of time. Also, using stemming as pre-processing before computing form-based TF-IDF scores is a standard practice. We use the [11] implementation of stemming in our code.

Lemmatization Although stemming provides a faster alternative, it is not always possible to work with stemmed output. This is because stemming often results in words which are out-of-dictionary, being a rule based system. For

instance, dance may be stemmed to danc by a stemmer. Though this would still maintain the same form-based similarity as before, a semantic similarity measure cannot work with out of dictionary words, since it needs to query an external knowledge source, such as WordNet. Hence, we need lemmatization, which maps words to their actual root forms(which themselves are valid words), before we compute semantic similarity. Also, stemming cannot handle cases such as 'be' and 'was', where the root itself undergoes change due to tense.

Once we perform pre-processing, we need to compute the pairwise similarity between sentences. We compute similarity in two ways - cosine similarity between tf-idf word vectors, and semantic similarity based on WordNet.

Cosine similarity After stemming, we represent each sentence as a vector, where every index corresponds to a word. Each element is the TF-IDF(term frequency inverse document frequency) score for that sentence corresponding to the word at that index. We use the following definitions of term frequency and inverse document frequency

- **Term Frequency**
  We normalize the frequency of a term in a sentence by the **maximum frequency of any word in that sentence**. Un-normalized versions of term frequency are often biased towards repetition of words. Also normalizing by sentence length, sentences which use a diverse set of words get unfairly penalized. Moreover, longer sentences do tend to have slightly more information content. Hence, we do not want to entirely remove the influence of length by normalizing based on it.

$$tf_S(w_i) = \frac{frequency_S(w_i)}{\max_i frequency_S(w_i)} \tag{3}$$

- **Inverse Document Frequency**
  The inverse document frequency is computed as follows

$$idf_S(w_i) = \log(\frac{N}{n_{w_i}}) \tag{4}$$

After converting each sentence to its tf-idf vectors, we compute the pairwise similarity using the cosine similarity measure.

Semantic Similarity There are various measures of Wordnet-based similarity, which can be classified loosely as

- Path-based Measures such as Wu-Palmer and Leacock-Chodorow
- Information content based measures, such as Lin
- Others, such as Lesk(based on word glosses)

We experimented with the different measures listed as examples above. However, we are unable to incorporate semantic similarity into our final evaluation, due to the prohibitive cost of lemmatization, which is an unavoidable step.

Setting Significance Threshold Since we first compute similarity for all pairs, if we admit an edge between every pair in our sentence graph, the the graph will end up being complete($O(n^2)$). This is inconvenient for the following reasons

- Many algorithms for community detection, influence maximization etc, are specifically designed for sparse graphs, since sparsity is a common attribute of real world networks. It becomes infeasible to use such approaches on a complete graph, since it is overtly dense.
- Edges with very low weight can introduce spurious correlation between sentence nodes in the sentence graph. This can reduce the final quality of our summarization.

Hence, we only admit edges whose weight(i.e pairwise similarity) is above a certain significance threshold. Though we vary this threshold and observe the effects on the sentence graph, for most of our experiments we maintain it in the range [0.01,0.2]. We once again notice a power law distribution, with very few edges having significant weight.

3.2 Community Detection

Most community detection algorithms attempt to optimize a measure of closeness or cohesion between individual communities. Though there are several mathematical formulations of a community [5], one of the most popular measures used is modularity. Here, Q denotes the modularity, m denotes the number of edges, A_{ij} is the element of the adjacency matrix, k_i is the degree of node i, k_j is the degree of node j, and the function δ_{ij} is 1 if both i and j belong to the same community, being 0 otherwise.

$$Q = \frac{1}{2m} \sum_i \sum_j A_{ij} - \frac{k_i k_j \delta_{ij}}{2m} \tag{5}$$

The above method is for unweighted graphs, though the extension to weighted graphs is trivial.

We use the Clauset-Newman-Moore algorithm for community detection, which performs greedy optimization of modularity. Starting with each node as an individual community, the algorithm iteratively merges communities in a greedy fashion, until no more increase in modularity can be achieved. This algorithm is a classical algorithm for community detection, and we choose this since our goal is not to compare different community detection methods, but to demonstrate the efficacy of performing community detection to observe inherent divisions in the sentence graph.

3.3 Budget Allocation

Our summary will have an upper bound on length specified either in terms of percentage of the document sentence count, or alternatively in number of

sentences to be extracted. Sometimes, summaries are also specified with an upper bound on the total number of characters or words. For the sake of simplicity, we work in a sentence limit setting. However, the method is easily extensible to the other settings as well.

Since we want to choose the influential nodes in each community separately, we need to distribute the summary's allowed quota of sentences between the communities. Each community's share is also called the budget, since this is the budget for influence maximization in that community. The scheme we use in our experiments is as follows

- Find the fraction of nodes in each community relative to the number of nodes in the entire graph. Retain only those communities into the set of admitted communities.
- For each admitted community, allot a budget equal to the fraction of nodes relative to the number of nodes in all the allotted communities.

3.4 Influence Maximization

We perform influence maximization under the linear threshold model for diffusion, which was one of the two diffusion models proposed in [4], the other being Independent Cascade. In this model, a node v is influenced by a node w with a weight $b_{v,w}$, such that

$$\sum_{w \ active \ neighbour \ of \ v} b_{v,w} \leq 1 \tag{6}$$

Each node then randomly chooses a threshold θ_v. If the sum of incoming weights from active neighbours exceeds this threshold, then the current node is considered to be activated. However, the MC simulations that are run to estimate the spread(since the thresholds are chosen randomly, we need to average over sufficiently many runs), are often quite expensive. $SIMPATH$([3]) is an algorithm which overcomes this limitation by restricting the spread estimation to a DAG, for which it can be computed in linear time. The simple algorithm makes O(n*k) calls to the spread estimation procedure. SIMPATH cuts down these calls significantly, and hence makes the usage of influence maximization more tractable.

4 Experiments and Results

4.1 Dataset

We are using TIPSTER's Computation and Language (cmp-lg) corpus [14] for evaluation purposes. This corpus contains 183 documents from cmp-lg collection and has been marked up in XML. The documents are scientific papers which appeared in Association for Computational Linguistics (ACL) sponsored conferences.

To use it for text summarization, for each paper, we extracted all the text present in its abstract. This is used as the ground truth or the reference summary. We run our summarization algorithms on the text extracted from the body. References, equations, footnotes and images are ignored. We also consider only the papers that have abstracts with length more that 5 sentences. This is done to ensure that the summary lengths are not too small. This leaves us with 20 papers in our dataset.
We have set our summary size to be 10 sentences, which is close the number of sentences that are present in an abstract.

4.2 Evaluation Metric

We used ROUGE-1 as our evaluation metric. ROUGE is a recall-based metric for fixed-length summaries which is based on n-gram co-occurrence. ROUGE-1 report a score based on the 1-gram matching between the ground truth summaries and the summaries to be evaluated. We chose ROUGE-1 because this has been shown to agree with human judgements the most by [6].

$$ROUGE - 1(s) = \frac{\sum_{r \in R} < \phi(r), \phi(s) >}{\sum_{r \in R} < \phi(r), \phi(r) >} \tag{7}$$

where R is the set of reference summaries and s is the summary that needs to be evaluates.

4.3 Baselines

We are using three baselines to evaluate our algorithms. Two of them are graph-based methods that run on the whole sentence graph.

PageRank We run the weighted version of PageRank algorithm on the whole sentence graph. We then select the top-k nodes which have the highest PageRanks as the summary.

Influence Maximization We run SimPath on the whole sentence graph with a budget of 10 sentences. The sentences returned by the algorithm are selected as the summary.

Centroid-based We also use the baseline used in [1] called a centroid-based method, which is a non-graph based technique. In this method, each word that appears in the text is assigned a centroid score. This is simply the product of its tf in the whole document and its idf. The centroid score of a sentence is simply the summation of the centroid scores of the words contained in it. We then choose the top-k sentences which have the highest centroid scores as the summary.

4.4 Parameter Tuning

Varying the significance threshold captures an important tradeoff between removing spurious edges and retaining significant ones. Too low a significant threshold results in a dense graph with spurious edges, reducing the quality of the summary produced. Too high a significance threshold results in pruning of significant edges and destroys the topical structure represented by the sentence graph. We need to find the optimal value of the threshold through empirical tuning of the threshold. For our experiments, we find this threshold to be 0.05 for the $TIPSTER$ dataset.

In practice, one would need to first tune this threshold on a validation set. However, due to paucity of data, we are unable to accommodate different validation and test sets for parameter tuning. The variation in ROUGE-1 scores for the various techniques, on varying the significance threshold can be seen in the graph below. We can see that the optimal point for most of the techniques is close to 0.05

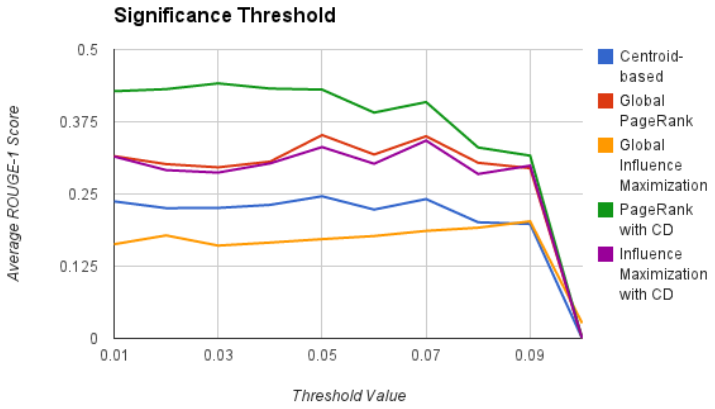

Fig. 1. Significance Threshold

5 Observations

5.1 Qualitative Analysis of Communities

1. Power Law Behaviour: We see that the number of significant communities is exponentially lesser than the number of non-significant ones. In other words, there are very few communities with a large number of nodes.

2. The number of significant communities is always upper-bounded by a small constant. This shows that the number of latent concepts(or threads of perception) in a document that humans can perceive is upper-bound by a small number, as observed in [1].
   Below, we have plotted the number of significant communities for the sentence graphs of some famous literary classics such as *Gift Of The Magi*, *Julius Caesar*, *Pride and Prejudice*.

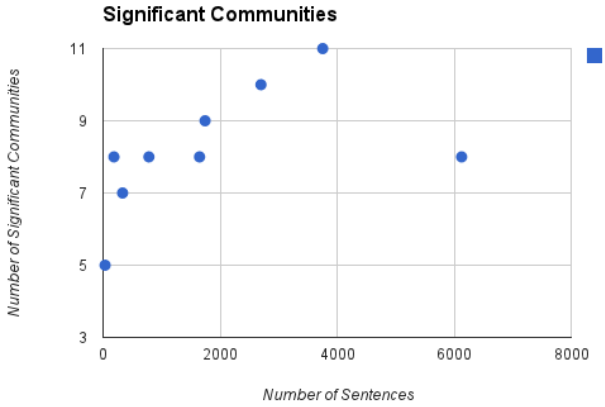

Fig. 2. Number of Significant Communities

5.2 Directionality of the Graph and Influence Maximization

We observe that community detection followed by influence maximization works best when the sentence graph is constructed in a forward directed fashion, and even performs better than community detection followed by PageRank, and PageRank on the entire graph for forward directed graphs. We hypothesize that this could be because of two reasons

1. Directed graphs are more suited for influence maximization because the spread of influence in influence maximization is modelled in an inherently directed fashion. In an an undirected graph, where every pair of nodes can either influence each other or not influence each other at all, it becomes difficult to model the circularity in the simulations for $SIMPATH$
2. Forward directed graphs suit influence maximization much more than backward directed graphs in the context of this dataset, since they accurately model the flow of information in the documents, each of which is a research paper. In a research paper(excluding the abstract) modelled by a forward directed graph, the sentences in the initial sections of the paper(such as the

Introduction) transmit their information to the latter sections(such as Algorithms or Experiments). Thus the influential sentences are biased to be from the initial sections of the paper. This is consistent with the way research papers are generally written, with the introduction giving a high level overview of the paper, and latter sections dealing with specific components.
In contrast, a backward directed graph will have the influential nodes biased to be from the latter sections. However, these sections are actually about specific components, rather than emphasizing on covering the entire breadth of the paper. Hence, the flow of influence is constrained to be in a direction opposite to the natural information flow.

Technique	***Avg. ROUGE-1 Score***		
	Undirected	*Forward*	*Backward*
Centroid-based	0.2460614741	0.2681227782	0.259555576
Global PageRank	0.3520220046	0.3859656948	0.372220496
Global Influence Maximization	0.1718369654	0.2342572809	0.220899580
PageRank with CD	0.430706545	0.3919357786	0.4335365905
Influence Maximization with CD	0.3312306809	0.4029592461	0.3779037538

Table 1. ROUGE-1 Scores

The complete set of readings (which include minimum and maximum scores) can be found at [10].

6 Conclusion and Future Work

6.1 Conclusion

We conclude that the approaches with Community Detection perform better than their global counter parts. Community Detection plays its role in discovering suppressed topics, and making sure that they have sufficient representation in the final summary. Moreover, there are only a few significant communities to be discovered in a document's sentence graph, and the number of such communities remains fairly constant across the size of the document.

Also, directionality of graph can have significant implications for the performance of influence maximization based methods, and has to be tailored to the requirements of the domain. In our case, we saw that forward directionality was favourable for the domain under consideration, i.e research papers.

6.2 Future Work

1. Using distributional similarity to model pairwise similarity between sentences. This will allow us to relate cases like the occurrence of the words **ship** in one sentence and **sea** in the other.

2. Experimenting with different schemes of budget allocation. For instance, we can adopt a regularized scheme of budget allocation(motivated by approaches such as L1 Regularized Logistic Regression) in Machine Learning. This prevents very large communities from getting a disproportionate share of the summary.
3. Extending our approaches to the multi-document summarization setting, with both meta-summarization and complete graph summarization approaches to handle multiple documents.

References

1. G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, Dec. 2004.
2. N. Garg, B. Favre, K. Riedhammer, and D. Hakkani-Tr. Clusterrank: a graph based method for meeting summarization. In *INTERSPEECH*, pages 1499–1502. ISCA, 2009.
3. A. Goyal, W. Lu, and L. V. S. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 211–220, Washington, DC, USA, 2011. IEEE Computer Society.
4. D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.
5. J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 631–640, New York, NY, USA, 2010. ACM.
6. C.-Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 71–78, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.
7. R. Mihalcea and P. Tarau. A language independent algorithm for single and multiple document summarization. In *In Proceedings of IJCNLP2005*.
8. R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In D. Lin and D. Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.
9. A. Nenkova and K. McKeown. A survey of text summarization techniques. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 43–76. Springer US, 2012.
10. S. of readings. http://goo.gl/cfmkwn.
11. M. Porter. Porter stemmer, 2000.
12. D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, Dec. 2002.
13. J. Sjöbergh and K. Araki. Extraction based summarization using a shortest path algorithm. In *Proceedings of the 12th Annual Natural Language Processing Conference NLP2006*, pages 1071–1074, Yokohama, Japan, 2006.
14. TIPSTER. Computation and language (cmp-lg) corpus, May 2003.

Appendix: A Qualitative Example

We show here the results from two of our techniques- Community Detection followed by PageRank and Community Detection Followed by Influence Maximization. The abstract is considered as the human or ground truth summary. We can see that one of the automatic summaries agrees with the human summary on the first sentence, which is a significant indicator of agreement.

Abstract

The formalism of synchronous tree-adjoining grammars , a variant of standard tree-adjoining grammars -LRB- TAG -RRB- , was intended to allow the use of TAGs for language transduction in addition to language specification .In previous work , the definition of the transduction relation defined by a synchronous TAG was given by appeal to an iterative rewriting process .The rewriting definition of derivation is problematic in that it greatly extends the expressivity of the formalism and makes the design of parsing algorithms difficult if not impossible . We introduce a simple , natural definition of synchronous tree-adjoining derivation , based on isomorphisms between standard tree-adjoining derivations , that avoids the expressivity and implementability problems of the original rewriting definition . The decrease in expressivity , which would otherwise make the method unusable , is offset by the incorporation of an alternative definition of standard tree-adjoining derivation , previously proposed for completely separate reasons , thereby making it practical to entertain using the natural definition of synchronous derivation . Nonetheless , some remaining problematic cases call for yet more flexibility in the definition ; the isomorphism requirement may have to be relaxed . It remains for future research to tune the exact requirements on the allowable mappings .

Community Detection Followed by PageRank

The formalism of synchronous tree-adjoining grammars , a variant of standard tree-adjoining grammars -LRB- TAG -RRB- , was intended to allow the use of TAGs for language transduction in addition to language specification. Synchronous TAGs specify relations between language pairs ; each language is specified with a standard TAG , and the pairing between strings in the separate languages is specified by synchronizing the two TAGs through linking pairs of elementary trees. This paper concerns the formal definitions underlying synchronous tree-adjoining grammars. In this paper , we describe how synchronous TAG derivation can be redefined so as to eliminate these problems. However, because certain derivations in the rewriting sense have no analogue under the new definition , some linguistic analyses may no longer be statable . A synchronous TAG G will be given as a set of triples where the Li and Ri are elementary trees , both initial and auxiliary , forming two component TAGs and , and is the linking relation between tree addresses in Li and Ri . Choose a link between two nodes in the current derived tree pair. Thus , the tree pair labeled pairs a

noun phrase -LRB- -RRB- initial tree dominating the proper noun ‘ John ’ with a logical term -LRB- T -RRB- dominating the constant john.

Community Detection Followed by Influence Maximization

Similarly , the pertinent entry under ‘ hopefully ’ specifies the translation of ‘ -LSB- hopefully -RSB- it wo n’t rain ’ as ‘ on espre qu’il ne va pas pleuvoir ’ , providing implicitly the subderivation mapping of ‘ hopefully ’ in its presentential position with ‘ on espre que ’ . The increased flexibility from the ability to perform such multiple adjunctions makes it conceivable to entertain using the natural definition of synchronous derivation . In previous work , links were typically thought of as impinging on the top of a node unless otherwise stated . Further flexibility can be obtained by allowing each link to specify whether it links to the top or bottom of the nodes . Thus , the link relation in a triple can be thought of as being of type , where dom -LRB- A -RRB- is the set of tree addresses in the tree A and and serve as markers to specify whether the link impinges on the top or bottom , respectively , of the specified node . The observation that synchronous TAGs under the new definition should be reducible to MCTAG was brought to our attention by Owen Rambow . The French critic example , however , remains problematic. The relation between the critic and the NP which it is semantically related to seems to be potentially unbounded.