Sentence Compression via Clustering of Dependency Graph Nodes

Ayman R. El-Kilany  
Faculty of Computers and Information, Cairo University  
Cairo, Egypt  
a.elkilany@fci-cu.edu.eg

Samhaa R. El-Beltagy  
Nile University  
Cairo, Egypt  
samhaa@computer.org

Mohamed E. El-Sharkawi  
Faculty of Computers and Information, Cairo University  
Cairo, Egypt  
m.elsharkawi@fci-cu.edu.eg

Sentence compression is the process of removing words or phrases from a sentence in a manner that would abbreviate the sentence while conserving its original meaning. This work introduces a model for sentence compression based on dependency graph clustering. The main idea of this work is to cluster related dependency graph nodes in a single chunk, and to then remove the chunk which has the least significant effect on a sentence. The proposed model does not require any training parallel corpus. Instead, it uses the grammatical structure graph of the sentence itself to find which parts should be removed. The paper also presents the results of an experiment in which the proposed work was compared to a recent supervised technique and was found to perform better.

Keywords: Sentence Compression; Summarization; Unsupervised Model; Dependency Graph.

1. Introduction

Sentence compression is the task of producing a shorter form of a single given sentence, so that the new form is grammatical and retains the most important information of the original one (Jing, 2000). The main application of sentence compression is within text summarization systems.

A lot of models have been developed for sentence compression with different methods and learning techniques, but the focus of most is on selecting individual words to remove from a sentence. Most of the existing models for sentence compression are supervised models that rely on a training set in order to learn how to compress a new sentence. These systems require large training sets, the construction of which requires extensive human effort. Unsupervised models on the other hand,
In this paper, we propose a new unsupervised model for sentence compression. Rather than considering individual words, phrases or even dependency graph nodes as candidates for elimination in the sentence compression task as previous work has done, the proposed model operates on the level of a data chunk in which related nodes of a sentence’s dependency graph are grouped together. The basic premise on which this work builds is that grouping related nodes in one chunk will most likely result in a chunk whose nodes are less related to other nodes in other chunks and that the removal of one of these chunks will have minimal effect on the overall sentence given appropriate constraints in the selection of the chunk or chunks to be removed. Our model follows a two stage procedure to reach a compressed version of a sentence. In the first stage, the sentence is converted into a group of chunks where each chunk represents a coherent group of words acquired by clustering dependency graph nodes of a sentence. Initial chunk generation is done through the use of the Louvain method (Blondel, Guillaume, et al, 2008) which hasn’t been used for sentence dependency graph node clustering before. Chunks are then further merged through a set of heuristic rules. In the second stage, one or more chunks are removed from sentence chunks to satisfy the required compression ratio such that the removed chunks have the least effect on sentence. Effect is measured using language model scores and word importance scores.

The rest of this paper is organized as follows: Section 2 discusses work related to the sentence compression problem followed by a brief description of the Louvain method in Section 3. A detailed description of the proposed approach is given in Section 4. Section 5 describes the experimental setup while Section 6 presents the results. Section 7 analyzes the results and section 8 presents the conclusion.

2. Related Work

In this paper, we develop an unsupervised model for sentence compression based on graph clustering. Before presenting our algorithm, we briefly summarize some of the highly related work on sentence compression.

The model developed by (Jing 2000) was one of the earliest works that tackled the sentence compression problem. Given that a sentence consists of a number of phrases, the model has to decide which phrases to remove according to many factors. One of the main factors is grammar checking that specifies which phrases are grammatically obligatory and can’t be removed. Also, the lexical relations between words are used to weigh the importance of the word in sentence context. Another factor is the deletion probability of a phrase which shows how frequently the phrase was deleted in compressions produced by humans. This deletion probability is estimated from a sentence compression parallel corpus.
In contrast to (Jing 2000), a noisy-channel model (Knight and Marcu, 2002) depends exclusively on a parallel corpus to build the compression model. The noisy-channel model is considered the baseline model for comparing many supervised approaches that appeared later. Given a source sentence \( x \), and a target compression \( y \), the model consists of a language model \( P(y) \) whose role is to guarantee that the compression output is grammatically correct, a channel model \( P(x|y) \) that captures the probability that the source sentence \( x \) is an expansion of the target compression \( y \), and a decoder which searches for the compression \( y \) that maximizes \( P(y)P(x|y) \). The channel model is acquired from a parsed version of a parallel corpus. To train and evaluate their system (Knight and Marcu, 200) used 1035 sentences from the Ziff–Davis corpus for training and 32 sentences for testing.

Another supervised approach is exemplified by the model developed by (Sporleder and Lapata 2005) which utilizes discourse chunking. According to their definition, discourse chunking is the process of identifying central and non-central segments in the text. The model defines a supervised discourse chunking approach to break the sentence into labeled chunks (which in this context is different than the definition we’ve given to a chunk) before removing non-central chunks in order to compress a sentence.

(Nomoto 2009) presents a supervised model for sentence compression based on a parallel corpus gathered automatically from the internet. The model generates candidate compressions by following defined paths in the dependency graph of the sentence where each path represents a candidate compression. A candidate compression gets classified to determine if it would work as a compression or not. The classifier used for this process was trained using 2,000 RSS feeds and their corresponding ‘sentence pairs’ gathered automatically from internet news, and was tested using 116 feeds. Nomoto was able to prove that his system significantly outperforms what was then the “state-of-art model intensive [sentence compression] approach known as Tree-to-Tree Transducer, or T3” (Nomoto 2009) proposed by (Cohn and Lapata, 2009).

As for unsupervised approaches, (Hori and Furui 2004) proposed a model for automatically compressing transcribed spoken text. The model operates on the word level where each word is considered as a candidate for removal either individually or with any other word in the sentence producing candidate compressions. For each candidate compression, the model calculates a language model score, a significance score and a confidence score then chooses the one with the highest score as its output.

Another word-based compression approach was developed by (Clarke and Lapata 2008). The approach views the sentence compression task as an optimization problem with respect to the number of decisions that have to be taken to find the best compression. Three optimization models were developed to represent the sentence compression problem (unsupervised, semi-supervised, supervised ) in the presence of linguistically motivated constraints, a language model score and a significance score are used to improve the quality of taken decisions.
(Filippova and Strube 2008) present an unsupervised model for sentence compression based on the dependency tree of a sentence. Their approach shortens sentence by pruning sub-trees out of a dependency tree. Each directed edge from head $H$ to word $W$ in dependency tree is scored. They score each edge with a significance score for word $W$ and a probability of dependencies which is the conditional probability of the edge type given the head word $H$. Probabilities of dependencies were gathered from a parsed corpus of English sentences. They built an optimization model to search for the smallest pruned dependency graph possible with the highest score. In their work they were able to prove that their system can outperform that of another un-supervised approach which was presented by (Clarke and Lapata, 2008).

Following the same strategy of (Nomoto 2009) by gathering resources automatically from news resources, (Cordeiro, Dias and Brazdil 2009) presented an unsupervised model which automatically gathers similar stories from internet news in order to find sentences sharing paraphrases but written in different lengths. The model then induces compression rules using alignment algorithms used in bioinformatics such that these rules would be used later to compress new sentences.

3. The Louvain Method

The Louvain method (Blondel, Guillaume, et al, 2008) is an algorithm for finding communities in large social networks. Given a network, the Louvain method produces a hierarchy out of the network where each level of the hierarchy is divided into communities and where nodes in each community are more related to each other than nodes in other communities. Community quality is measured using modularity. Modularity measures the strength of division of a network into communities (also called groups, clusters or modules). Networks with high modularity have dense connections between the nodes within communities but sparse connections between nodes in different communities.

The Louvain method has two steps that are repeated iteratively. Given a graph of $n$ nodes, each node is said to represent a community. All nodes are processed in order such that each node either joins one of the communities of its neighbors or remains in its current community according to the highest modularity gained. This step is performed iteratively until a local maximum of modularity is gained. Second, once a local maximum has been reached, the method builds a new graph whose nodes are the generated communities. The weight of the edges between new nodes in the new graph is the total weight of the edges between the nodes of old communities. After the new graph is built, steps one and two are repeated on this new graph until there are no more changes and maximum modularity is gained.
4. The Proposed Approach

We propose an approach for the sentence compression task. Our goal is to cluster related dependency graph nodes in a single chunk, and to then remove the chunk which has the least significant effect on a sentence. Towards this end, we follow six steps to reach a compression for the sentence:

1. **Chunking**: In this step, the dependency graph of a sentence is generated and then the nodes of dependency graph are grouped into chunks. The chunking process is done mainly through clustering the dependency graph nodes of a sentence using the Louvain method.

2. **Chunk Merging**: in this step, chunks are further refined using a set of heuristic rules that make use of grammatical relations between words in a sentence to enhance the quality of the resulting chunks.

3. **Word scoring**: In this step, each word in a sentence is assigned an importance score. The word scores are used in later steps to calculate the importance of words in candidate compressions relative to a generated dependency graph.

4. **Candidate compression generation**: in this step, a number of candidate compressions are generated based on obtained chunks.

5. **Candidate compression scoring**: here each of the candidate compressions generated from step 4, is scored.

6. **Compression generation**: in this step, the compression which best fits a desired compression ratio, is produced.

Each of these steps is detailed in the following subsections.

4.1. **Chunking**

Given an input sentence, the goal of this phase is to turn this sentence into a number of chunks where each chunk represents a coherent group of words from the sentence. To accomplish this step, the dependency graph of the sentence is first generated then the Louvain method is used over the dependency graph of the sentence. Louvain method is used because of its ability to find related communities/chunks inside the network/graph of nodes (that in our case represent the sentence) regardless of the size of the network (big or small).

The Louvain method was chosen over other graph clustering algorithms for two main reasons. First, the nature of a dependency graph itself where there isn’t a defined distance measure between the nodes of the graph and where only links between nodes define a relationship. Since the Louvain method is used on large networks that have the same features and which primarily makes use of links between nodes, it is an excellent candidate for what we want to do. Second, the Louvain method strategy of finding the best local cluster for a node from its direct neighbors clusters before finding global clusters is suited for finding coherent chunks of words inside the
dependency graph since the nodes with the most number of links between each other in a dependency graph should represent a coherent chunk.

The Stanford parser (de Marneffe, MacCartney, and Manning 2006) was used to generate the dependency graph of a sentence using the parser’s basic typed dependencies. As for the Louvain method, the C++ implementation provided by the authors was used.

Since the Louvain method produces a hierarchal decomposition of the graph, we had the choice to either use the lowest or highest level of the hierarchy, where the highest level yields fewer chunks. We use the lowest level of the hierarchy if the number of resulting chunks is ≤ 5; otherwise, the highest level of hierarchy is used. It’s important to note that this value was obtained empirically by experimenting with a dataset of 100 randomly collected internet news sentences.

For the sentence "Bell, based in Los Angeles, makes and distributes electronic, computer and building products" clustering results obtained from the first level of the hierarchy are illustrated in Fig.1 where nodes enclosed in each circle represent a generated chunk. To our knowledge, the Louvain method has not been used in this context before.

![Fig. 1: Example of clustering results for a dependency graph](image)

### 4.2. Chunk Merging

The Louvain method is a general purpose algorithm that does not address linguistic constraints; it just works on dependencies between nodes in a graph as pure relations without considering the grammatical type of relationships between the nodes/words. So, the method can easily separate a verb and its subject by placing them in different chunks making it hard to produce a good compression for the sentence later.
To solve this problem, we had to do some post-processing over the resulting chunks using the grammatical relations between nodes/words from dependency graph to improve their quality. Towards this end, some linguistic rules that preserve verb/object/subject relation in all its forms (clausal subject, regular subject, passive subject, regular object, indirect object, clausal compliments with external/internal subject...) were crafted. For example one of these rules dictates that if there is a verb in a chunk and if its subject is in another chunk, then both chunks should be merged.

Another rule ensures that all conjunctions, propositions and determiners are attached to the chunks that contain words to their right in the natural order of sentence. Since such types of words affect just the words beside them, there is no need to distribute them across chunks. For example, if we have the following sentence: "Bell, based in Los Angeles, makes and distributes electronic, computer and building products.", then the proposition "in" will be placed in the same chunk as the noun "Los", the conjunction "and" will be placed in the same chunk as the verb "distributes" and the other conjunction "and" will be in the same chunk with the noun "building".

Chunk merging is not used if merging leads to extreme cases. Extreme cases are those in which all chunks are merged into one or two chunks or where one chunk contains more than 80% of the original sentence. In such cases the merging conditions are relaxed and the highest level of the hierarchy is used.

When this step was applied on the experimental dataset described in section 5, the average chunk size was 8.4 words ±5.1 words.

4.3. Word Scoring

We chose a scoring scheme that reflects the importance of words in the context of the sentence such that a word that appears at a high level within a dependency graph and has many dependents should be more important than other words at lower levels with less dependents.

Using the structure of the dependency graph where the root of the graph gets a score=1, we score each word in the sentence – except stop words – according to the following equation:

$$S(W) = \frac{cc(W)}{1 + cc(W) + \sum_{i=1}^{n(w)} cc(w(i))} \times S(p(W)) \quad (1)$$

Where $S$ is the score, $W$ is the word, $CC$ is the child count of the word within the dependency tree, $p(W)$ is the parent of the word $W$ within the dependency tree, $n(w)$ is the total number of siblings of word $w$, $W(i)$ is the word for sibling $i$, in the dependency graph.
According to Eq. (1), each word is scored according to its level in the dependency graph and the number of children it has relative to the number of children of its siblings in the dependency graph. The score of the word is a percentage of the score of its parent in the dependency graph which means that the deeper the level of the word is in the dependency graph, the lower its score. The number of children of the word relative to its siblings' number of children defines the percentage of the score the word would get from its parent which means a word that has more children should get a better score than a sibling with less number of children.

4.4. Candidate Compression Generation

Candidate compressions are generated out of chunks obtained from the previously described steps. For example, if the chunking and merging steps produced 3 chunks (Chunk1, Chunk2, Chunk3) then different combinations from these chunks constitute the candidate compressions (Chunk1 – Chunk2 – Chunk3 – Chunk1Chunk2 – Chunk1Chunk3 – Chunk2Chunk3). Words in each combination will be placed in their natural order in the original sentence. Since one of these combinations shall be the resulting compression of our system, we forced a new rule to ensure the quality of the model’s output. The rule forces all combinations to include the chunk that has the root word of the dependency graph. The root of the dependency graph has the most important word in the sentence as all other words in the sentence depend on it.

Given the previous example, if Chunk2 is the one containing the root of the dependency graph then the generated combinations would be limited to: (Chunk2 – Chunk1Chunk2 – Chunk2Chunk3).

For example, candidate compressions for the sentence "Bell, based in Los Angeles, makes and distributes electronic, computer and building products" giving that the chunk that has the root node is "Bell makes and distributes products" would be as follow:

- Bell makes and distributes products
- Bell, based in Los Angeles, makes and distributes products
- Bell, makes and distributes electronic, computer and building products

4.5. Candidate Compression Scoring

Given the candidate compressions that were produced in the previous subsection, one of them needs to be selected as an output. To do so, each candidate compression is scored with a language model score and an importance score. The candidate compression with the highest combined score is the one to be produced as an output.

For the language model, we used a 3-gram language model provided by (Keith Vertanen) that was trained using the newswire text provided in the English Gigaword corpus with a vocabulary of the top 64k words occurring in the training text.
Using the 3-gram language model, we scored each candidate compression according to the following equation:

Given a candidate compression of \( n \) words:

\[
LMScore(CandidateCompression) = \log[P(W_0 | Start)] + \log[P(\text{end} | W_n, W_{n-1})] + \sum_{i,j} (\log[P(W_{i+1} | W_i, W_j)] + \log[P(W_i | W_j, W_{j-1})])
\]

Where \( i, j \) are for the two words that represent end of a chunk and start of following chunk for every joining between two different chunks in the candidate compression.

Contrary to the usual usage of a language model, we use it only to find the probability of the start word of the compression, end word of the compression and adjacent words in the compression that represent the connection of different chunks even if these words follow the natural order within a sentence.

For Example: If a candidate compression \( X \) has the following words from the original sentence: \( w_0, w_1, w_2, w_3, w_6, w_7, w_8, w_9, w_10 \) where \( w_0, w_1, w_2, w_3 \) are in chunk 1 and \( w_6, w_7, w_8, w_9, w_10 \) are in chunk 3

\[
LMScore(X) = \log[P(W_0 | Start)] + \log[P(\text{end} | W_{10}, W_9)] + \log[P(W_7 | W_6, W_3)] + \log[P(W_6 | W_3, W_2)]
\]

In this way, candidate compressions with few chunks are given a better LMScore than other candidate compressions.

The importance score for a candidate compression \( X \) that has \( n \) words is calculated using the following equation:

\[
\text{ImportanceScore}(X) = \sum_{k=1}^{n} \text{Score}(W_k)
\]

Where \( W_k \) is the importance score of word \( k \) calculated using Eq. (2) presented in section 4.3

4.6. Compression Generation

The model now has a number of candidate compressions for an input sentence and each of these has an LMScore and an ImportanceScore. Given the required compression ratio range, we search among candidate compressions for one that satisfy the desired compression range. Then we choose the compression that maximizes: \( \text{LMScore} + \text{ImportanceScore} \).
For example, given a desired compression range between 0.6 to 0.69, the resulting compression for the sentence "Bell, based in Los Angeles, makes and distributes electronic, computer and building products" will be "Bell, based in los Angeles, makes and distributes products" as it will get highest score.

It is possible that there will be no compression which satisfies the required compression ratio. In such cases, the original sentence will be returned with no compression.

5. Experimental Setup

The goal of our experiment was to prove that our unsupervised algorithm can achieve at least comparable results with one of the highly-rated supervised approaches in the domain of sentence compression. Towards this end, we’ve compared our results to those of the free approach model developed by (Nomoto 2009) through manual and automatic evaluation.

We chose to compare our approach to a supervised one which utilizes dependency graphs for sentence compression since supervised approaches are usually trained for a specific dataset which means that they should be highly competent in carrying out their task on this specific dataset. So, by selecting a reputable supervised model, and demonstrating comparable or superior performance, we can assert the validity of our approach.

In our evaluation, we tried to build a fair evaluation environment as suggested by (Napoles, Van Durme and Burch 2011) in their work about best conditions to evaluate sentence compression models.

Nomoto kindly provided us with the dataset he used to evaluate his system as well as with the output of his system at different compression ratios. We evaluated our model’s output and his model’s output from the dataset on the same sentences and same compression ratios; so we considered 0.6 and 0.7 compression ratios only. It is important to note that using both our approach and Nomoto’s free approach, it is impossible to get an exact compression ratio, so for example a desired compression ratio of 0.6 will be matched with any compression that falls in the range of 0.6 to 0.69.

For the compression ratio of 0.6 we evaluated 92 sentences and 90 sentences for the compression ratio of 0.7. It's important to note that these numbers represent what both models succeeded to compress at the specified compression ratios.
Sentences that our model didn't succeed to compress at the specified compression ratios were ignored. The reason our model failed to compress certain sentences to a specific ratio is based on the fact that our system removes chunks, not words. So adjusting the compression to a specific ratio is not always possible. The RSS feeds of the original sentences were considered as the gold standard.

For manual evaluation, scores were solicited from two human judges who defined their English level as fluent. The human judges followed the scoring guidelines shown in Table 1 which are the same guidelines used for evaluating Nomoto’s approach. Intelligibility means how well the compression reads. Representativeness indicates how well the compression represents its source sentence.

A tool was developed specifically for the judges to collect their evaluation scores. Through this tool the judges were given the original sentence as well as various compressions generated by the proposed system, by Nomoto’s system and the RSS feed. The judges were unaware of which compression was generated by which model in order to eliminate any bias on their part.

As for automatic evaluation, we used the parsing-based evaluation measure proposed by Riezler et al. (2003) where grammatical relations found in a compression are compared to grammatical relations found in the gold standard to find precision, recall and F1-score. We calculated precision, recall and F1-score for our model and Nomoto's model on both 0.6 and 0.7 compression ratios against RSS feeds (gold standard). We used the Stanford parser to find grammatical relations in both compressions and the gold standard.

<table>
<thead>
<tr>
<th>Meaning</th>
<th>Explanation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Bad</td>
<td>Intelligibility: means the sentence in question is rubbish, no sense can be made out of it. Representativeness: means that there is no way in which the compression could be viewed as representing its source.</td>
<td>1</td>
</tr>
<tr>
<td>Poor</td>
<td>Either the sentence is broken or fails to make sense for the most part, or it is focusing on points of least significance in the source.</td>
<td>2</td>
</tr>
<tr>
<td>Fair</td>
<td>The sentence can be understood, though with some mental effort, it covers some of the important points in the source sentence.</td>
<td>3</td>
</tr>
<tr>
<td>Good</td>
<td>The sentence allows easy comprehension; it covers most of important points talked about in the source sentence.</td>
<td>4</td>
</tr>
<tr>
<td>Excellent</td>
<td>The sentence reads as if it were written by human; it gives a very good idea of what is being discussed in the source sentence.</td>
<td>5</td>
</tr>
</tbody>
</table>
6. Results

6.1. Manual Evaluation Results

Evaluation scores for each evaluation criterion and each model were collected from the tool and averaged.

Table 2 shows how our model and the free approach model performed on the dataset along with the gold standard. Table 3 shows the differences between results of our evaluation of the free approach model and the results reported in the Nomoto (2009) paper.

<table>
<thead>
<tr>
<th>Model</th>
<th>CompRatio</th>
<th>Intelligibility</th>
<th>Representativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold Standard</td>
<td>0.69</td>
<td>4.71±0.38</td>
<td>3.79±0.76</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>0.64</td>
<td>4.09±0.79</td>
<td>3.57±0.66</td>
</tr>
<tr>
<td>Free Approach</td>
<td>0.64</td>
<td>3.34±0.78</td>
<td>3.25±0.65</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>0.74</td>
<td>4.15±0.73</td>
<td>3.78±0.59</td>
</tr>
<tr>
<td>Free Approach</td>
<td>0.73</td>
<td>3.63±0.78</td>
<td>3.52±0.66</td>
</tr>
</tbody>
</table>

Table 3: Results of our manual evaluation for Free Approach model and gold standard and evaluation obtained from paper by Nomoto.

<table>
<thead>
<tr>
<th>Model</th>
<th>Intelligibility</th>
<th>Representativeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our judges</td>
<td>Nomoto’s judges</td>
<td>Our judges</td>
</tr>
<tr>
<td>Gold Standard (0.69)</td>
<td>4.71</td>
<td>4.78</td>
</tr>
<tr>
<td>Free Approach (0.64)</td>
<td>3.34</td>
<td>3.06</td>
</tr>
<tr>
<td>Free Approach (0.74)</td>
<td>3.63</td>
<td>3.33</td>
</tr>
</tbody>
</table>

When reading table 3, it is important to note that Nomoto’s judges evaluated a slightly bigger subset, as there are cases as previously mentioned before, where our system failed to produce a result at a specified compression. The difference between the results of our evaluation for the free approach model and the results reported in the original paper was as follows: intelligibility has a mean difference of +0.2 and representativeness has a mean difference of +0.1. So our judges actually gave a
slightly higher score to compressions produced by Nomoto’s system, than his own evaluators. All in all however, the scores are very close.

Table 2 shows that our model performed better than the free approach model on intelligibility and representativeness but with higher rates on intelligibility.

We ran the t-test on the results of the evaluation scores obtained from both our model and Nomoto’s (shown in table 2) to find out if the difference between our model and Nomoto's model is significant or not. The t-test revealed that the difference over both criteria and both compression ratios is statistically significant as all resulting p-values for the t-test were less than 0.01.

Also, our model for the 0.7 compression ratio, managed to achieve close results to those of the gold standard dataset especially on the representativeness ratings.

### 6.2. Automatic Evaluation Results

Table 4 shows how our model and the free approach model compare against the RSS feeds gold standard on 0.6 and 0.7 compression ratios in terms of precision, recall and F1-score. The results show that our model performed better than the free approach model.

<table>
<thead>
<tr>
<th>CompRatio</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Approach</td>
<td>0.64</td>
<td>46.2</td>
<td>43</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>0.64</td>
<td>50.1</td>
<td>47.5</td>
</tr>
<tr>
<td>Free Approach</td>
<td>0.73</td>
<td>50.3</td>
<td>50.8</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>0.74</td>
<td>51.9</td>
<td>53.7</td>
</tr>
</tbody>
</table>

To summarize, the results show that our proposed model manages to achieve better results than the free approach model on two different compression ratios using two different evaluation methods.

Table 5 shows examples of compressions created by our model, Nomoto's model, gold standard compressions and relevant source sentences.

### 6.3. Results Analysis

We believe that the reason that our model performed better than Nomoto's model on intelligibility can be attributed to the way we used the dependency graph.
Table 5: Examples from our model output, Nomoto’s model output and gold standard

<table>
<thead>
<tr>
<th>Source</th>
<th>Gold</th>
<th>Nomoto’s</th>
<th>Our Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>Officials at the Iraqi special tribunal set up to try the former dictator and his top aides have said they expect to put him on trial by the end of the year in the deaths of nearly 160 men and boys from Dujail, all Shiites, some in their early teens.</td>
<td>Iraqi officials expect to put Saddam Hussein on trial by the end of the year in the deaths of nearly 160 men and boys from Dujail.</td>
<td>Officials at the Iraqi special tribunal set up to try the former dictator by the end of the year in the deaths of nearly 160 men and boys from Dujail.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Microsoft’s truce with Palm, its longtime rival in palmtop software, was forged with a rare agreement to allow Palm to tinker with the Windows Mobile software, the companies’ leaders said here Monday.</td>
<td>Microsoft’s truce with Palm, its longtime rival in palmtop software, was forged with a rare agreement to allow Palm to tinker with the Windows Mobile software.</td>
<td>Microsoft’s truce with Palm was forged with a rare agreement to allow Palm to tinker with the Windows Mobile software.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>The six major Hollywood studios, hoping to gain more control over their technological destiny, have agreed to jointly finance a multimillion-dollar research laboratory to speed the development of new ways to foil movie pirates.</td>
<td>The six major Hollywood studios have agreed to jointly finance a multimillion-dollar research laboratory to speed the development of new ways to foil movie pirates.</td>
<td>The six major Hollywood studios, over their technological destiny, have agreed to jointly finance a multimillion-dollar research laboratory to speed to foil movie pirates.</td>
</tr>
</tbody>
</table>

Nomoto’s model finds its candidate compressions by selecting paths inside the dependency graph from the root node to a terminal node which means that words inside the compression can be scattered all over the graph which won’t guarantee high intelligibility. Our model on the other hand, searches for densities inside the dependency graph and then chooses some to keep and others to discard which allows more grammatical compressions than Nomoto’s model.

As for performing better on representativeness, the fact that we are searching for densities inside the dependency graph to keep or to discard, will usually result in discarded chunks that bear little relevance to the rest of the chunks. This contrasts to...
Nomoto's model which considers that the words of the compression are scattered all over dependency graph nodes.

Automatic evaluation results support the superiority of our model against Nomoto's model as it shows that our model’s output is more similar to the gold standard than Nomoto's model thus re-enforcing the results of manual evaluation.

7. Conclusion

This paper presented a method for sentence compression by considering the clusters of dependency graph nodes of a sentence as the main unit of information. Using dependency graphs for sentence compression is not novel in itself as both (Nomoto 2009) and (Filippova and Strube 2008) have also employed it for the same task. However clustering dependency graph nodes into chunks and eliminating chunks rather than nodes does present a new approach to the sentence compression problem. Comparing this work to that of a supervised approach that employ dependency graph as we do has shown that the presented model performs better when compressing sentences both in terms of manual and automatic evaluation.

Future work will consider incorporating the proposed model into a more general context like paragraph summarization or multi-documents summarization. We also plan on experimenting on other datasets.

Acknowledgments

The authors wish to thank Professor Tadashi Nomoto for kindly availing his experimental dataset and results for use by the authors.

References