An approach to Abstractive Text Summarization

Huong Thanh Le
Hanoi University of Science and Technology
Hanoi, Vietnam
huonglt@soict.hut.edu.vn

Tien Manh Le
Hanoi University of Science and Technology
Hanoi, Vietnam
tienlehut@gmail.com

***Abstract*—Abstractive summarization is the technique of generating a summary of a text from its main ideas, not by copying verbatim most salient sentences from text. This is an important and challenge task in natural language processing. In this paper, we propose an approach to abstractive text summarization based on discourse rules, syntactic constraints, and word graph. Discourse rules and syntactic constraints are used in the process of generating sentences from keywords. Word graph is used in the sentence combination process to represent word relations in the text and to combine several sentences into one. Experimental results show that our approach is promising in solving the abstractive summarization task.**

Keywords- abstractive text summarization, discourse relation, word graph

I. INTRODUCTION

Automatic text summarization is the technique which automatically creates an abstract or summary of a text. It gained widespread interest due to overwhelming amount of textual information available in electronic format. Text summarization techniques can be broadly grouped into abstractive summarization (AS) and extractive summarization (ES). Most research on text summarization are ES [2,11] since it is easier and faster than AS. ES extracts verbatim most salient sentences from text. Meanwhile, AS is relied on Natural Language Processing (NLP) techniques to copy-paste sentence fragments from the input document and maybe combine the selected content with extra linguistic information in order to generate the final summary. There are two main problems with ES. First, the textual coherence is not guaranteed as resolving anaphora resolution is not paid attention in this approach. Second, redundant phrases still exist in the summary. AS can solve this problem by carrying out NLP techniques to post-process the output of ES such as sentence truncation, aggregation, generalization, reference adjustment and rewording [4,6,8]. However, AS is still a major challenge for NLP community despite some work on sub-sentential modification [4,6].

Recent approaches in AS use word graphs to represent a document [3,9]. These graphs are then used to produce document abstracts, allowing the algorithm to compress and merge information. Representing documents by word graphs is a new and potential approach for generating abstractive summary. However, this approach still has many problems, as discussed in Section 2. In this paper, we concentrate on rhetorical structure and word graph to generate an abstractive summary. Several strategies are proposed to solve existing problems with word graph based approaches. The input of our system is an extractive summary after anaphora resolution. That means all pronouns have been replaced by corresponding nouns/noun phrases (NPs).

The rest of this paper is organized as follows. Section 2 analyzes existing problems with the word graph and proposes our strategies to deal with them. Our sentence reduction's method is introduced in Section 3. Section 4 presents our method of merging sentences using word graph. Experimental results are discussed in Section 5. Finally, Section 6 concludes the paper and gives some insight for future work.

II. CONSTRUCTING GRAPH

A word graph consists of nodes and edges. Existing approaches on AS [3,9] use nodes to store information about words and theirs POS tag and edges to represent adjacency relations between word pairs. A new sentence is generated by connecting all words in a path of the word graph.

The approaches using word graph for single document summarization still have problems as many sentences with incorrect meaning can be generated. This is because the generation algorithms find paths among words on the graph, regardless of their syntactically correctness and the original text. An example of sentences with incorrect meaning is shown below.

Example 1: $Mẹ_{mother}$ $Bách_{Bach}$ mua_{buy} $thuốc_{medecine}$ $về_{back}$ cho_{for} $uống_{drink}$. Sau_khi_{after} $uống_{drink}$, $Bách_{Bach}$ $có_{has}$ $biểu\_hiện_{symptom}$ $đỏ_{red}$ $môi_{lip}$ $và_{and}$ $nổi_{appear}$ $bọng_{bubble}$ $nước_{water}$ $ở_{at}$ tay_{hand} $và_{and}$ $chân_{leg}$.

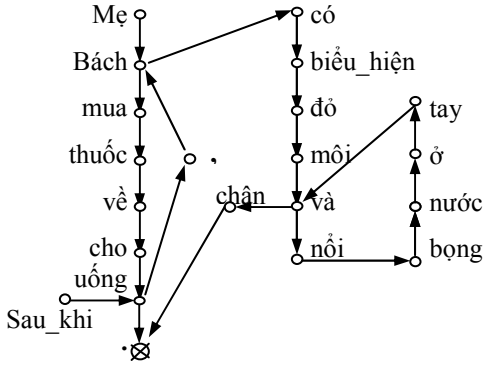

Figure 1. The word graph representation for Example 1

In Fig. 1, each small circle represents a word of the text; the symbol ⊗ means the end of a sentence. Each arrow is created by connecting two adjacent words in a sentence. The above word graph can generate sentences "$Bách_{Bach}$ mua_{buy}

thuốc$_{medecine}$ *về*$_{back}$ *cho*$_{for}$ *uống*$_{drink}$" and "*Mẹ*$_{mother}$ *Bách*$_{Bach}$ *có*$_{has}$ *biểu_hiện*$_{symptom}$ *đỏ*$_{red}$ *môi*$_{lip}$ *và*$_{and}$ *nổi*$_{appear}$ *bọng*$_{bubble}$ *nước*$_{water}$ *ở*$_{at}$ *tay*$_{hand}$ *và*$_{and}$ *chân*$_{leg}$", which do not reflex the correct meaning of the text. In addition, *"đỏ môi và chân*$|_{red\ lip\ and\ leg}$*"* should not be generated since it does not reflex the correct meaning of the original text. Pagerank method or adding information about sentence position used in [3] and [9] cannot deal with this problem.

The problem of incorrect meaning of a new phrase does not only happen with NPs, but also with other phrases such as verb phrases (VPs) and adjective phrases (AdjPs). In order to solve this problem, instead of finding paths containing keywords using scores or shortest paths as in [3] and [9], our abstractive summary generation process is separated into two stages: sentence reduction and sentence combination. The sentence reduction step is based on input sentences, keywords of the original text and syntactic constraints. Word graph is used only in the sentence combination stage.

The problem of incorrect meaning of a new phrase is solved in the sentence reduction stage using two strategies. First, all basic phrases[1] (including basic NPs, basic VPs, and basic AdjPs) from the extractive sentences that contain keywords are used as essential materials for the sentence reduction stage. A new sentence is created by connecting the first phrase to the last one in the original sentence and then expanding its left and right sides to satisfy syntactic constraints. A detailed description of this procedure is introduced in Section 3.

In the above example, to generate a new sentence from the original sentence "*Mẹ*$_{mother}$ *Bách*$_{Bach}$ *mua*$_{buy}$ *thuốc*$_{medecine}$ *về*$_{back}$ *cho*$_{for}$ *uống*$_{drink}$.", the basic NP in this sentence that contains the keyword "*Bách*" is "*Mẹ Bách*". Therefore, "*Mẹ Bách*" (not "*Bách*") is used as the subject of this sentence. To generate a new sentence from the original sentence "Sau_khi$_{after}$ uống$_{drink}$, Bách$_{Bach}$ có$_{has}$ biểu_hiện$_{symptom}$ đỏ$_{red}$ môi$_{lip}$ và$_{and}$ nổi$_{appear}$ bọng$_{bubble}$ nước$_{water}$ ở$_{at}$ tay$_{hand}$ và$_{and}$ chân$_{leg}$.", "*Mẹ Bách*" cannot be the subject of the new sentence since it does not appear in the original sentence.

The second strategy to solve the first problem is to consider stop words, prepositions, numerals, auxiliary words, and negative words (e.g., "không$_{not}$", "chẳng$_{never}$") as separated nodes. Otherwise, the real meaning of the sentence could be changed when generating new sentences.

The second problem of existing approaches using word graph is that ungrammatical sentences can be generated by the word graph. Let us consider the example below.

Example 2: Hà_Giang$_{HaGiang}$ trở_thành$_{becomes}$ điểm$_{place}$ hấp_dẫn$_{attractive}$ khách$_{guess}$ du_lịch$_{tourist}$ trong$_{inside}$ và$_{and}$ ngoài$_{outside}$ nước$_{country}$. Sở$_{department}$ Văn_hóa$_{culture}$ và$_{and}$ Du_lịch$_{tourist}$ Hà_Giang$_{HaGiang}$ đã$_{has}$ ký$_{sign}$ hợp_tác$_{cooperation}$ phát_triển$_{develop}$ du_lịch$_{tourist}$ với$_{with}$ Sở$_{department}$ Văn_hóa$_{culture}$ và$_{and}$ Du_lịch$_{tourist}$ của$_{of}$ nhiều$_{many}$ thành_phố$_{city}$.

[1] The Vietnamese chunker, created by Nguyen Le Minh and Cao Hoang Tru, belongs to the VLSP project http://vlsp.vietlp.org:8080/demo/?page=home is used for extracting basic phrases from sentences.

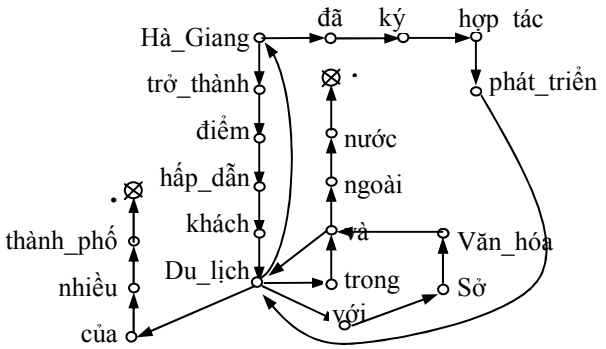

Figure 2. The word graph representation for Example 2

The sentence "*Sở*$_{department}$ *văn_hóa*$_{culture}$ *và*$_{and}$ *ngoài*$_{outside}$ *nước*$_{country}$." is generated by the above word graph. However, this is only a NP, not a sentence. Moreover, this NP is an incorrect name of a Vietnamese department. This is because the word "*và*$_{and}$" at the middle of the original department name having two output branches in the word graph: one to the rest of the department name and one to another branch. This sentence is created by visiting the second branch from the word "*và*$_{and}$" in this case. This sentence never appears in our system since this is the case of incorrect meaning and is solved by our two strategies in the sentence reduction stage mentioned above.

Another drawback of existing word graph based approaches is that these researches do not care about word/phrase meaning. Different words/phrases that refer to the same concept are represented as different nodes in their graphs. As a result, sentences that contain these nodes cannot be merged to create a new sentence with richer information than the old ones. To solve this problem, an anaphora resolution module[2] has been integrated into our ES system. Then the output of our ES system is used as the input of our abstractive summarizer. This is different than that of [3] and [9] in which anaphora resolution has not been considered. From such a type of input, all nodes that refer to a concept are grouped into one. That is, the text field of a node will store multi-values, as illustrated in Fig. 3. If the original sentence uses one value in this text field, the system can use any value in this group to generate a new sentence. We consider two cases: (i) synonym words; and (ii) different expressions refer to the same concept.

To deal with the first case, a synonym dictionary is used. For example, "*phát_biểu*$_{say}$" and "*tuyên_bố*$_{declare}$" are two synonyms, they are just considered as one node in the graph.

The second case is solved by using coreference resolution. For example, if the original sentence is "*Vũ_Du*$_{VuDu}$ *dùng*$_{use}$ *thuốc*$_{medecine}$ *Biseptol*." and by coreference resolution, we know that *"bệnh_nhân*$_{patient}$*", "Vũ_Du*$_{VuDu}$*", "Du*$_{Du}$*",* and *"bệnh_nhân*$_{patient}$ *Vũ_Du*$_{VuDu}$*"* refer to the same object, the original word graph in Fig. 3a can be expanded as in Fig. 3b. Such of coreference

[2] Due to the scope of this paper, the anaphora resolution step is not mentioned in this paper.

resolution's rules are proposed by us and are integrated in our system.

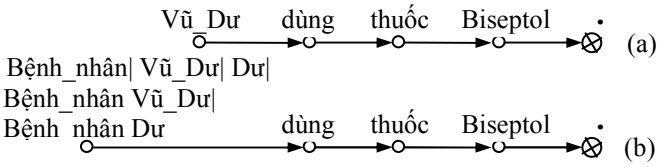

Figure 3. The graph representation for the sentence "*Vũ_Dư dùng thuốc Biseptol.*"

If the next sentence involves "*Vũ_Dư*" such as "*Bệnh_nhân*$_{patient}$ *bị*$_{suffer\ from}$ *biến_chứng*$_{side\_effect}$ *nặng*$_{heavy}$", the word "*bệnh_nhân*$_{patient}$" is also mapped to the first node of the graph as in Fig. 3b.

Our graph to represent the input text is organized as follow. The graph G = (V, E) consists of a set of vertexes (nodes) V and a set of edges E. A vertex keeps four kinds of information:

- a text field stores words or phrases that refer to a concept;
- a POS field stores the grammatical role of the text field. If the text field has several values, the largest POS tag will be assigned.

An edge connects two vertexes in the graph. Two vertexes are connected if their texts are adjacent in the input text.

The input of the algorithm to create a graph is the original document and its extractive summary. The extractive summary has been tokenized, POS tagged, solved coreferences and defined unsplittable phrases. The words and unsplittable phrases are called textual units of the input text. The output of the algorithm is a graph G = (V, E) represents the extractive summary. Steps to generate the graph that represents the extractive summary of a text is shown below:

- Detect all phrases that refer to a proper name in the original text (e.g., "*bệnh_nhân*$_{patient}$", "*Vũ_Dư*$_{VuDu}$", "*Dư*$_{Du}$", and "*bệnh_nhân*$_{patient}$ *Vũ_Dư*$_{VuDu}$"). These phrases are called unsplittable phrases.
- For each textual unit from sentences in the extractive summary:
  - Add a vertex v_i corresponding to this textual unit when: (i) the textual unit is a stop word, prepositions, numerals, and negative words; or (ii) the textual unit with its POS does not exist in the graph. If the textual unit is a proper name or an unsplittable phrase, add all coreference words/phrases of this textual unit to the text field of the new node.
  - Create a directed edge by connecting the vertex corresponding to the previous textual unit with the vertex corresponding to the considered textual unit.

As mentioned earlier in Section 2, our process of generating an abstractive summary is divided into two stages: sentence reduction and sentence combination. The stage of sentence reduction is introduced next.

III. SENTENCE REDUCTION

The input of sentence reduction module is the extractive summary of the document and keywords of the original document. Keywords are extracted from the original document by computing its tf-isf (term frequency - inverted sentence frequency) and getting top k per cent keywords with highest tf-isf. The optimal value of k in our system is 15% and it is determined by our experiments. The output of this module is another version of summary that is shorter than its input text.

By studying how humans write summaries, Jing and McKeown [6] found that professional abstractors often reuse the text in an original document, and then edit the extracted sentences for producing the summary. Applying this idea, instead of creating new sentences from keywords, we locate important phrases in original sentences (basic phrases in the original sentences that contain keywords) and use them as essential materials for generating an abstractive summary. This method permits us reduce ungrammatical phrases and produce sentences whose meaning are close to the original sentences.

To create a new sentence from important phrases of the original sentence, the fragment that spans from the first important phrase to the last one in the original sentence is generated. This fragment is considered as an essential part in the original sentence. Then other words of the original sentence are added to the beginning and end of this fragment to create a syntactically correct sentence whose meaning of the original sentence is still remained.

The process of generating a sentence from the essential fragment of a sentence is divided into two steps: (i) completing the beginning of a sentence; and (ii) completing the end of a sentence.

A. Completing the end of a sentence

The input of this process is the original sentence that has been divided into basic phrases (NP, VP, etc.) and the essential fragment of the original sentence. To investigate grammatical problems with fragments generated by our system, we carried out an experiment using a data set of 200 documents collected from online newspapers. Experimental results shown that fragments end with the following phrases cannot be the end of a syntactically correct sentence:

- The fragment ends with a NP, which can be the subject of a clause/sentence, or the object of the main VP of the original sentence.
- The fragment ends with a VP, which follows by a NP or an AdjP in the original sentence; or the VP ends with a verb (V) and follows by a preposition phrase in the original sentence.

Based on our observation, the process of filling the end of a sentence is as follow:

- If the fragment ends with a NP and there is an AdjP or a VP right after that at the original sentence, connect that AdjP or VP to the end of the fragment.
- The fragment ends with a VP and there is a NP or an AdjP or a VP right after that at the original sentence,

connect that NP or AdjP or VP to the end of the fragment.

- The fragment ends with a VP. That VP ends with a verb and follows by a preposition phrase in the original sentence. In this case, the preposition phrase is connected to the end of the fragment.

B. Completing the beginning of a sentence

The input of this process is the original sentence that has been divided into basic phrases (NP, VP, etc.) and the essential fragment of the original sentence after completing the end of a sentence. By investigating fragments returned by our experimental results, we found out that the NP and the VP at the beginning of the fragments may not be the main NP or the main VP of the original sentence. This is because keywords of the sentence may be in the object of the main verb of the sentence; or in the preposition phrase of the sentence.

Finding the subject or the main verb of the sentence by locating the first NP or the first VP of the sentence, respectively, is not always correct since these phrases can be at the adVP of the sentence. Finding these phrases by locating the NP or the VP right before these phrases is not always correct either. Therefore, filling the beginning of the fragment is more complicate than filling the end of the fragment.

By studying written text, we found that the important parts of a sentence are often located at the beginning of the sentence. Therefore, when creating a new sentence from the essential fragment of the original sentence, the beginning of the fragment is expanded to the beginning of the original sentence. After that, some rules are applied to remove unimportant parts at the beginning of the new fragment. To recognize these unimportant parts, rules to detect discourse relations at the sentence-level [8] are applied. In order to understand rules to detect discourse relations at the sentence-level, Rhetorical Structure Theory (RST) is introduced next.

1) Rhetorical Structure Theory

Rhetorical Structure Theory (RST) [10] is a method of representing the coherence of text. It models the rhetorical structure of a text by a hierarchical tree that labels discourse relations between spans. This hierarchical tree diagram is called a "rhetorical tree" or "discourse tree". The leaves of an RST tree correspond to elementary discourse units (edus), which are clauses or clause-like units with independent functional integrity, whereas the internal tree nodes correspond to larger spans.

Fig. 4 represents the discourse tree of Example 3. Instead of displaying the full text of each tree node, we cite the first and last edus that contribute to it (e.g., "3.1-3.2", "3.1-3.3"). An internal tree node contains one or several names (e.g., elaboration, explanation) of the discourse relations that hold between adjacent, non-overlapping spans. The span that participates in a discourse relation is either a *nucleus* (N) or a *satellite* (S). The nucleus plays a more important role than the satellite in respect to the writer's intention. If both spans have equal roles, they are both considered as *nuclei* in the relation.

Example 3: [You should meet Thanh today$_{3.1}$] [after you finish this work$_{3.2}$]. [He will go to Saigon tomorrow.$_{3.3}$]

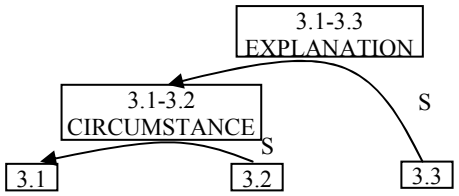

Figure 4. The Discourse Tree of Example 3

To construct the discourse structure of a text, the following tasks should be performed: (i) segmenting text into edus; (ii) recognizing discourse relations between spans; and (iii) constructing a discourse tree that represents the discourse structure of the text.

Most of research on RST for English bases on cue phrases such as *because, but, although,* etc. to segment text [12]. For example, the sentence *"*We cannot be sure the product is safe *although* we have tested it." can be splitted into two edus *"*We cannot be sure the product is safe" and "*although* we have tested it.", based on the cue phrase *although*. In addition to cue phrases, syntactic information is also used in [8] to segment text into edus.

Researchers have defined many discourse relations such as list, sequence, elaboration, cause, result, evidence, etc. These relations are divided into three types: N-N, N-S, S-N. In this research, since we only concern in remove unimportant part at the beginning of sentences, only S-N relations are concerned. Identifying names of discourse relations and constructing the discourse tree of the text are out of scope of this research.

The next section will introduce our method of recognizing S-N relations from sentences and removing unimportant part at the beginning of a sentence.

2) Removing unimportant part at the beginning of a sentence

The first step of this stage is to recognize S-N relations from sentences. Based on this, sentence reduction is done by keeping the N part in the summary.

As mentioned in Section 3.2.1, text segmentation can be done by using cue phrases [12] and syntactic information [8]. As far as we know, there is no Vietnamese syntactic parser whose accuracy is higher than 90%. Therefore, it is not reliable to use the output of syntactic parser for the text segmentation task. The segmentation process for Vietnamese cannot rely simply on cue phrases neither, as analyzed below.

Since Vietnamese is a monosyllabic language, a cue phrase may be recognized incorrectly as a part of another word. A word may also be recognized incorrectly as a cue phrase. Let us consider Example 4 below:

Example 4:

a. Tôi$_I$ rất$_{very}$ buồn$_{sad}$ khi$_{when}$ em$_{you}$ không$_{did\ not}$ đến$_{come}$.

b. Chăng_mấy_khi$_{rarely}$ anh$_{you}$ đến$_{come\ to}$ nhà$_{house}$ tôi$_{my}$.

In Example 4a, the word "khi$_{when}$" is a cue phrase. In Example 4b, "khi" is a part of the word "chăng_mấy_khi$_{rarely}$", and it is not a cue phrase. To deal with this problem, information about cue phrases is

combined with information about words and their POS tag to detect cue phrases in a given sentence[3].

The list of cue phrases is created by our empirical research on Vietnamese text and by inheriting cue phrases and its template from [5,8,12]. Examples of our template using cue phrases are:

Bởi_vì$_{since}$ S nên$_{therefore}$ N.

Nếu$_{if}$ S thì$_{then}$ N.

In general, the strategy of sentence reduction is language independent. However, the process of filling the end of a sentence is language dependent since each language has its own grammar principles.

IV. SENTENCE COMBINATION

After sentence reduction, the process of sentence combination is carried out. By studying how humans write summaries, we found that the following cases can be merged to create a new sentence with richer information:

a. Two short and consecutive sentences with the same subject:

<sentence 1> = <noun|NP> <VP 1>

<sentence 2> = <noun|NP> <VP 2>

Two sentences are considered as consecutive if they are adjacent in the extractive summary. Two sentences have the same subject (i.e., <noun|NP>) if they start from the same node with the POS is a noun or a NP in the graph. The merged sentence in this case is

<new sentence> = <noun|NP> <VP 1> và$_{and}$ <VP 2>

b. A sentence has a component that provides more detailed information for a noun or a NP of the previous sentence. This sentence always starts with a phrase mentioned to the previous sentence such as "đó là$_{this\ is}$", "điều đó$_{this\ problem}$". The list of such phrases is manually created by our empirical research.

<sentence 1> = <left text 1> <noun|NP1> <right text 1>

<sentence 2> = <a phrase mentioned to the previous sentence > <left text 2> <NP2> <right text 2>

in which <NP2> starts with <noun|NP1> and contains proper name in its remaining part. In this case, an edge is created from the node corresponding to <NP2>, to the node corresponding to <right text 1> in the graph. If all keywords of <sentence 2> is in <NP2> only, the two sentences are merged into one:

<new sentence> = <left text 1> <NP2> <right text 1>

Example 5: Các_em$_{they}$ chỉ$_{only}$ ăn$_{eat}$ **cơm$_{rice}$** với$_{with}$ **muối$_{salt}$**. Đó$_{this}$ là$_{is}$ tình_cảnh$_{situation}$ cuộc_sống$_{life}$ của$_{of}$ các_em$_{they}$ **học_sinh$_{pupil}$ Trường$_{school}$ Mường_Lý$_{MuongLy}$**.

The graph representation for Example 5 is shown in Fig. 5. The second sentence in Example 5 starts with the phrase "*đó là$_{this\ is}$*" and contains the NP "*các_em$_{they}$ học_sinh$_{pupil}$ Trường$_{school}$ Mường_Lý$_{MuongLy}$*", which is a detailed description of the noun "*Các_em$_{they}$*" in the previous sentence. Since all keywords of the second sentence are in the NP "*các_em$_{they}$ học_sinh$_{pupil}$ Trường$_{school}$ Mường_Lý$_{MuongLy}$*", the two sentences in Example 4 are combined to create the new sentence "*các_em$_{they}$ học_sinh$_{pupil}$ Trường$_{school}$ Mường_Lý$_{MuongLy}$ chỉ$_{only}$ ăn$_{eat}$ cơm$_{rice}$ với$_{with}$ muối$_{salt}$*"

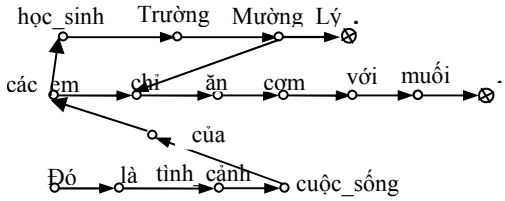

Figure 5. The graph representation for Example 5

c. A sentence has a component that provides more detailed information for a clause of the previous sentence.

<sentence 1> = <left text 1> <clause 1>

<sentence 2> = <left text 2> <component 2>

<component 2> in <sentence 2> starts with a phrase with similar meaning to <clause 1> in <sentence 1>. Notice that two consecutive sentences rarely use the same words to express a meaning, but synonyms are used instead. A synonym dictionary is created by us to detect such cases.

If all keywords of <sentence 2> is in <component 2>, the two sentences are merged into one.

<new sentence> = <left text 1> <component 2>

Example 6: "Mỹ$_{U.S.}$ đã$_{has}$ bày_tỏ$_{expressed}$ **lo_ngại$_{concern}$** về$_{about}$ mối **đe_dọa$_{threat}$** xâm_nhập$_{intrusion}$ **mạng$_{Internet}$** ngày_càng$_{day\ by\ day}$ gia_tăng$_{increasing}$" , ông$_{Mr.}$ **Hagel** phát_biểu$_{said}$. Điều$_{the\ problem}$ đáng$_{worth}$ chú_ý$_{attention}$ là$_{is}$ ông$_{Mr.}$ **Hagel** đã$_{has}$ đưa$_{issue}$ ra tuyên_bố$_{statement}$ ngay trước mặt$_{in\ front\ of}$ các đại_diện$_{representatives}$ của$_{of}$ chính_phủ$_{government}$ Trung_Quốc$_{Chinese}$ tại$_{in}$ Đối_thoại$_{dialogue}$ **Shangri - La** .

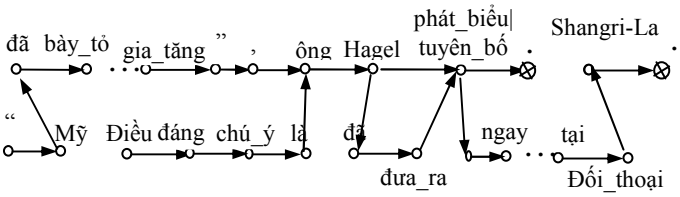

Figure 6. The graph representation for Example 6

In Example 6, the clause "*ông$_{Mr.}$ Hagel phát_biểu$_{said}$*" in the first sentence has the same meaning with "*ông$_{Mr.}$ Hagel đã$_{has}$ đưa$_{issue}$ ra tuyên_bố$_{statement}$*" in the second sentence. Therefore, these two sentences are combined to create a new sentence:

"Mỹ$_{U.S.}$ đã$_{has}$ bày_tỏ$_{expressed}$ lo_ngại$_{concern}$ về$_{about}$ mối đe_dọa$_{threat}$ xâm_nhập$_{intrusion}$ mạng$_{Internet}$ ngày_càng$_{day\ by\ day}$ gia_tăng$_{increasing}$" , ông$_{Mr.}$ Hagel đã$_{has}$ đưa$_{issue}$ ra tuyên_bố$_{statement}$ ngay trước mặt$_{in\ front\ of}$ các đại_diện$_{representatives}$ của$_{of}$ chính_phủ$_{government}$ Trung_Quốc$_{Chinese}$ tại$_{in}$ Đối_thoại$_{dialogue}$ Shangri - La .

The strategy of sentence combination is language independent.

[3] The softwares vnTokenizer and vnTagger, created by Le Hong Phuong (at http://mim.hus.vnu.edu.vn/phuonglh/softwares), are used for segmenting a Vietnamese text into words and tagging POS.

V. EXPERIMENTAL RESULTS AND DISCUSSION

As far as we know, there is no abstractive summarizing corpus for Vietnamese. Therefore, to carry out experiments with the summarizing system, we have to create a corpus by ourselves. Our corpus consists of 50 documents collected from several Vietnamese newspaper websites (e.g., Dantri, VnExpress, etc.) and belongs to two categories: economy and culture. The lengths of documents are various from 300 words to 1000 words. Each document has 22 sentences in average. The abstractive summaries were created manually by hand (one summary per document) with approximately 100 words in length.

The input of our abstractive summarizer is the output of our extractive one, which generates summaries with approximately 120 words in length. The output of our abstractive summarizer contains 100 words in average.

Among 433 sentences generated by our abstractive summarizer, 95% sentences are syntactic correct; 72% of those sentences are complete in meaning with unimportant parts at the end of sentences being removed. Most cases of incomplete sentences are due to the process of completing the end of a new sentence in the sentence reduction phrase. Reasons for this problem are:

- In the case of elaborative clauses situating between the main NP and the main VP of a sentence, the system misrecognizes the VP of the clause as the main VP of the sentence.
- The basic phrases of a sentence is detected incorrectly by the Vietnamese chunker, whereas information about basic phrases are the key point in completing the end of a new sentence.

The abstractive summaries generated by our system are also compared with the summaries in the corpus, using the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measurement [7]. The ROUGE measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary and the ideal summaries created by humans. In our experiments, since each document has only one summary, we only compare a candidate summary with a reference one.

Using the above formula, we get values of Rouge-1 and Rouge-2 of 0.2513 and 0.1344, respectively. Since there is no work on generating abstractive summaries using the same corpus with us, we cannot compare our experimental results with other research. However, according to [3], Rouge-1 and Rouge-2 values when comparing abstractive summaries created by two people are 0.3088 and 0.1069, respectively[4]. It indicates that our approach is promising in solving the text summarization task. However, since text generation in general and automatically abstractive text summarization in particular is still a challenge task, more work should be done to improve the quality of the system.

[4] The corpus used in [3] is different than ours.

VI. CONCLUSIONS AND FUTURE WORK

This paper has introduced an approach to abstractive text summarization, which consists of two stages: sentence reduction and sentence combination. The sentence reduction stage is based on discourse rules to remove redundant clauses at the beginning of a sentence, and syntactic constraints to complete the end of the reduced sentence. The sentence combination stage is based on word graph to present relations among words, clauses and sentences from the input text. New sentences that combine information from several sentences are generated by using word graph. Experimental results show that our approach is promising in solving the AS task.

To improve the system, our future works include: (i) propose methods to improve the meaning completeness of sentences generated in the sentence reduction phrase; (ii) propose methods to further compress sentences; and (iii) investigating strategies to efficiently combine sentences in the summary.

ACKNOWLEDGMENTS

This work was supported by the Vietnam Ministry project, under Grant B2012 – 01 - 24.

REFERENCES

[1] Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. Numerische Mathematik, vol. 1, pp. 269–271.

[2] Gunes, E. and Radev, D.R. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. J. Artif. Int.Res., 22(1):457–479.

[3] Ganesan, K., Zhai, C., Han, J. 2010. Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. In Proc. of Coling 2010, pages 340–348.

[4] Knight, K. and Marcu, D. 2000. Statistics-based summarization - step one: sentence compression. In Proc. of AAAI 2000.

[5] Hoang, T.P. 1980. Vietnamese grammar. Publisher of professional school.

[6] Jing, H. and McKeown, K. R. 2000. Cut and paste based text summarization. In Proc. of NAACL 2000.

[7] Lin, C.Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Proc.of NTCIR Workshop 2004.

[8] Le, H.T., Abeysinghe, G. and Huyck, C. 2004. Generating Discourse Structures for Written Texts. In Proc. of COLING 2004, Switzerland.

[9] Lloret, E., Palomar, M. 2011. Analyzing the Use of Word Graphs for Abstractive Text Summarization. In Proc. of IMMM 2011.

[10] Mann, W. C. and Thompson, S. A. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, vol. 8(3), 243-281.

[11] Mihalcea, R. and Tarau,P. 2004. TextRank: Bringing order into texts. In Proc. of EMNLP-04.

[12] Marcu, D. 1997. The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts", PhD Thesis, Department of Computer Science, University of Toronto.