SUMMARIZATION SYSTEM EVALUATION VARIATIONS BASED ON N-GRAM GRAPHS

GEORGE GIANNAKOPOULOS
NCSR DEMOKRITOS, GREECE
AND
VANGELIS KARKALETSIS
NCSR DEMOKRITOS, GREECE

Abstract. Within this article, we present the application of the AutoSummENG method within the TAC 2010 AESOP challenge. We further present two novel evaluation methods based on n-gram graphs. The first method is called Merged Model Graph (MeMoG) and it uses the n-gram graph framework to represent a set of documents with a single, "centroid" graph, offering state-of-the-art performance. The second method is called Hierarchical Proximity Graph (HPG) evaluation and it uses a hierarchy of graphs to represent texts, aiming to represent different granularity levels under a unified view. The experiments indicate that both novel methods offer very promising performance in different aspects of evaluation, improving on AutoSummENG scores.

1. Introduction

Throughout the history of artificial intelligence, there has existed the challenge of automatically analyzing text. For several years the summarization community has progressed and flourished, moving from single-document to multi-document summarization with promising results. However, the advancement of summarization systems implied a serious question: how can one measure automatically the quality of a single summary, or of a summarization system?

Automatic methods for the evaluation of summaries exist for some time now [HLZF05, Lin04, ZLMH06, SK09] and correlate highly to the measure of *responsiveness*. This measure, first appearing within the DUC community (see *e.g.,* [Dan05]), is expected to represent a measure of information completeness and linguistic quality for a given text, as a human assesses it. Even though the correlation of the automatic methods to the human grades is high on the system level, until recently there were some other desired characteristics that did not coexist in a single automatic method. More precisely:

- Language-neutrality. A method that does not require language dependent resources (thesauri, lexica, etc.) can be applied directly to different languages, without even the use of stemming or other language-specific preprocessing.
- Full automation. A method should not require human intervention, apart from the human model summaries.
- Context-sensitivity. A method should take into account contextual information, so that well-formedness of text is taken into account. Well-formedness can be loosely defined as the quality of a text that allows easy reading. A text that is a random sequence of words would lack this quality, even if the words are on topic.

The AutoSummENG method [GKVS08] (AUTOmatic SUMMary Evaluation based on N-gram Graphs), holds all these qualities, while bearing results with high correlation to the responsiveness measure, which indicates correlation to human judgment. Based on the knowledge gained by AutoSummENG, we hereby describe two novel variations using n-gram graphs: the Merged Model Graph (MeMoG) variation and the Hierarchical Proximity Graph (HPG) variation, offering the next step in n-gram graph based methods.

In the following paragraphs, we introduce the basic notions and algorithms for the representation of texts through n-gram graphs (Section 2). We then describe the methodology for the comparison, taking into account different representations that we use in our variations, and how this comparison leads to the grading of summaries and systems (Section 4). We present experimental results on the TAC2010 AESOP task (Section 5) and conclude the paper (Section 6) with the lessons learned.

2. System Overview

The AutoSummENG system [GKVS08] is based upon the JInsect library[1] of "n-gram graph"-based text processing. For our experiments in TAC 2010, we applied the AutoSummENG method, as well as two novel variations over the TAC 2010 AESOP task data. Thus, the study refers to three methods:

- The first method is the original AutoSummENG, for default parameters of $L_{\min}$, L_{MAX} and D_{win} for this year's corpus. This method creates an n-gram graph representation of the evaluated text and another n-gram graph per model summary. Then, the measure of Value Similarity is used to compare the similarity of the evaluated text to each model summary. The average of these similarities is considered to represent the overall performance of the summary text.
- The second method, instead of comparing the graph representation of the evaluated summary text to the graph representation of individual model texts and averaging over them, calculates the *merged* graph of all model texts. Then, it compares the evaluated summary graph to this overall model graph. We term this variation the Merged Model Graph method (MeMoG) and it aims to non-linearly combine the content of texts into one representative whole.
- The third method aims to alleviate the burden of estimating AutoSummENG parameters (see [GKVS08] for more). The approach uses graphs of minimal n-gram size (n=2) as a first level description of texts and, then, identifies subgraphs creating a second level of graphs. This second level of graphs, termed *proximity graph*, expresses neighborhood of subgraphs (viewed as distinct symbols) in the original text. Iteratively, new, higher levels of proximity graphs are then built using the subgraphs of the previous level as symbols to form a hierarchy of graphs, representing texts with multiple granularity levels.

In order to introduce the reader to the method and its alternatives, we need to recapitulate the basic concepts of AutoSummENG and the n-gram graph representation theory.

3. Representation and Basic Algorithms

In the domain of natural language processing, there have been a number of methods using *n-grams*. An n-gram is a, possibly ordered, set of words or characters, containing n elements (see Example 3.1). N-grams have been used in summarization and summary evaluation [BV04, LH03,

[1] See http://sourceforge.net/projects/jinsect and http://www.ontosum.org for more information.

CS04]. In the automatic summarization domain, n-grams mostly appear as *word* n-grams, as happens in the ROUGE/BE family of evaluator methods [HLZ05, Lin04].

Example 3.1. Examples of n-grams from the sentence: *This is a sentence.*
Word unigrams: this, is, a, sentence
Word bi-grams: this is, is a, a sentence
Character bi-grams: th, hi, is, s_, _a, ...
Character 4-grams: this, his_, _is_, ...

3.1. **Extracting N-grams.** To extract the n-grams (S^n) of a text T^l, we follow the (elementary) algorithm indicated as algorithm 1. The algorithm's complexity is linear to the size $|T|$ of the input text T.

Input: text T
Output: n-gram set SS^n
`// T is the text we analyze`
1 $SS^n \leftarrow \emptyset$;
2 **for** *all i in [1,length(T)-n+1]* **do**
3 | $SS^n \leftarrow SS^n \cup S_{i,i+n-1}$
4 **end**

Algorithm 1: Extraction of n-grams

The algorithm applies no preprocessing (such as extraction of spaces, punctuation or lemmatization). Furthermore, it obviously extracts overlapping parts of text, as the sliding window of size n is shifted by one position and not by n positions at a time. This technique is used to avoid the problem of segmenting the text. The redundancy apparent in this approach proves to be useful *similarly to a convolution function*: summing similarities over a scrolling window may prove useful when you do not know exactly where to start matching two strings.

In the case of summary evaluation we may compare common n-grams between a peer (judged) summary and a model summary. The extracted, overlapping n-grams are certain to metch corresponding n-grams of the model summary, if such n-grams exist. That would not be the case for a method where the text would be segmented in equally sized n-grams.

Example 3.2. *Application of our method to the sentence we have used above, with a requested n-gram size of 3 would return:*
{*'Do ', 'o y', ' yo', 'you', 'ou ', 'u l', ' li', 'lik', 'ike', 'ke ', 'e t', ' th', 'thi', 'his', 'is ', 's s', ' su', 'sum', 'umm', 'mma', 'mar', 'ary', 'ry?'*}
while an algorithm taking disjoint n-grams would return
{*'Do ', 'you', ' li', 'ke ', 'thi', 's s', 'umm', 'ary'*} *(and '?' would probably be omitted). The segmentation has reduced the number of existing n-grams examined, based on the disjointness prerequisite.*

The *n-gram graph* is a graph $G = \{V^G, E^G, L, W\}$, where V^G is the set of vertices, E^G is the set of edges, L is a function assigning a label to each vertex *and to each edge* and W is a function assigning a weight to every edge. The graph has n-grams as its vertices $v^G \in V^G$. The edges $e^G \in E^G$ (the superscript G will be omitted where easily assumed) connecting the n-grams indicate proximity of the corresponding vertex n-grams.

The edges can be weighted by the distance between the two neighboring n-grams in the original text, or the number of co-occurrences within a given window (as indicated below we use the co-occurrences for the TAC task). We note that the meaning of distance and window size changes

by whether we use character or word n-grams. The labeling function L for edges assigns to each edge the concatenation of the labels of its corresponding vertices' labels in a predefined order: for directed graphs the order is the order of the edge direction while in undirected graphs the order can be the lexicographic order of the vertices' labels. To ensure that no duplicate vertices exist, we require that the labeling function is an one-to-one function.

More formally:

Definition 3.3. *if* $S = \{S_1, S_2, ...\}, S_k \neq S_l$, *for* $k \neq l, k, l \in \mathbb{N}$ *is the set of* distinct *n-grams extracted from a text* T^l, *and* S_i *is the* i*-th extracted n-gram, then* $G = \{V^G, E^G, L, W\}$ *is a graph where* $V^G = S$ *is the set of vertices* v, E^G *is the set of edges* e *of the form* $e = \{v_1, v_2\}$, $L : V^G \to \mathbb{L}$ *is a function assigning a label* $l(v)$ *from the set of possible labels* $\mathbb{L}$ *to each vertex* v *and* $W : E^G \to \mathbb{R}$ *is a function assigning a weight* $w(e)$ *to every edge.*

In our implementation, the edges E are assigned weights of $c_{i,j}$ where $c_{i,j}$ is the number of times a given pair S_i, S_j of n-grams happen to be neighbors in a string within some distance D_{win} of each other. The distance between two strings in a text is the absolute difference of the positions of their first characters in the text. The vertices v_i, v_j corresponding to n-grams S_i, S_j that are located within this distance D_{win} are connected by a corresponding edge $e \equiv \{v_i, v_j\}$. In the TAC 2010 case we use the *symmetric approach* for *character n-gram* graph extraction, which has proved to be the most promising in several experiments [GKVS08].

3.2. **N-gram Graphs and Proximity Graphs.** A *proximity graph* (PG) is a graph that describes relations of proximity between *symbols* $s_i \in S$, where S the alphabet of symbols, in a domain where an operator of *proximity* P over $S \times S$ is defined. Thus, $P : S \times S \to \mathbb{R}$ gives a graded value for the proximity of two symbols within a world where proximity of symbols makes sense. In the summary evaluation case the world is the world of texts, and we define P as:

$$P(s_1, s_2) = |\{(s_i, s_j) : s_i = s_1, s_j = s_2 \text{ and } d(s_i, s_j) \leq D_{\text{win}}\}| \tag{1}$$

where $d(s_i, s_j)$ is the distance between s_i, s_j in the original text. P in fact measures the *number of times* two strings are found to be neighbors within the given distance D_{win} in a text. Given Equation 1, we can define a proximity graph as follows.

Definition 3.4. *If* $R = \{R_1, R_2, ...\}, R_k \neq R_l$, *for* $k \neq l, k, l \in \mathbb{N}$ *is the set of* distinct symbols *extracted from a text* T^l, *and* R_i *is the* i*-th extracted symbol, then* $J = \{V^J, E^J, M, X\}$ *is a graph where* $V^J = S$ *is the set of vertices* v, E^J *is the set of edges* e *of the form* $e = \{v_1, v_2\}$, $M : V^J \to \mathbb{L}$ *is a function assigning a label* $m(v)$ *from the set of possible labels* $\mathbb{M}$ *to each vertex* v *and* $X : E^J \to \mathbb{R}$ *is a function assigning a weight* $w(e)$ *to every edge.*

In fact, n-gram graphs are a special case of proximity graphs, because in n-gram graphs the set of symbols is the set of distinct n-grams: $R = S$. The weighting function on R is based on the cardinality of co-occurrences of n-grams within D_{win} distance of each other, as this is calculated by P. The generalized notion of proximity graphs, however, allows creating multiple levels of graphs, where the symbol set changes on each level.

Elaborating, consider a mapping $R_g : \mathbb{G} \to R$, where $\mathbb{G}$ is the set of possible n-gram (sub)graphs and R the set of symbols. Also, consider a proximity measure P', which measures the times two graphs $G_1, G_2 \in \mathbb{G}$ where found to be neighbors in a *sequence of graphs*, within a distance $D_{\text{win}}{}'$ of each other. Then, we can create proximity graphs of graphs. Exactly this notion of PGs of graphs is used in the following section to represent texts as a hierarchy of PGs, attempting to capture the information of multiple granularity in texts.

3.3. **Hierarchical Proximity Graphs.** A hierarchical proximity graph of $L \in \mathbb{N}*$ levels is a hierarchy H of proximity graphs, where subgraphs of symbols from a lower level $l-1$ form the symbols for the next upper level $l, l \in [1, L]$. Each level H_l holds a proximity graph J_l and an index of symbols I_l for that level.

To create the first level, we extract the set of *non-overlapping*[2] n-grams from the source text. Then, the index I_1 maps, through a bijection, every distinct n-gram in the source to an integer-symbol. This allows the conversion of the original text to a sequence Z of integers. Then, vertices are created in the proximity graph J_1: one for each symbol. Following the creation of vertices, and given a window D_{win}, all symbols the are found to be neighbors within a maximum distance of D_{win} in Z have their vertices linked by an edge. The distance between two n-grams n_1, n_2 is now measured as the number of n-grams (and not characters) between n_1 and n_2. The process of finding neighbors is repeated for every symbol in Z in the graph creation process.

Each subgraph of J_1 generated in every such iteration is called an *s-neighborhood*. Each s-neighborhood is considered to be a symbol for the next level of graphs and is, thus, mapped to an integer in the corresponding I_l index. This, from $l = 1$ we extract the s-neighborhoods that form the I_2 symbols. Then, the sequence of the current level is replaced by the sequence of s-neighborhoods.

For each of the following levels $1 < l \leq L$ of representation in the hierarchical graph the process is as follows.

- Get the s-neighborhood from the previous level $l-1$.
- Create the s-neighborhoods of the current level, adding them to I_{l+1} as symbols.
- Generate the current level PG J_l.

We note a few important design choices we have made:

- Every level uses a different size of D_{win}. In fact, on each level, $D_{\text{win}l} = \lfloor D_{\text{win}} * l \rfloor$ and $\lfloor \cdot \rfloor$ is the round-down operator. This stands upon the intuition that the notion of neighborhood when you go from a word to a paragraph changes completely: things that are further away are considered neighbors.
- The index allows fuzzy matching of PGs, with a parametrically defined fuzziness. We will not describe the process of the index creation due to space limitations.

Given the above algorithms, we can extract from a given text a hierarchy of PGs, that describe the text in a variety of levels. This hierarchy $\mathbb{J} =< J_1, ..., J_L >$ is the Hierarchical Proximity Graph (HPG).

3.4. **MeMoG: The Merged Model Graph.** Given two instances of n-gram graph representation G_1, G_2, there is a number of operators that can be applied on G_1, G_2 to provide the n-gram graph equivalent of union, intersection and other such operators of set theory. In our applications we have used the *update operator* U (similar to the merging operator), which allows the creation of a "centroid" graph. The update function $U(G_1, G_2, l)$ takes as input two graphs, one that is considered to be the pre-existing graph G_1 and one that is considered to be the new graph G_2. The function also has a parameter called the *learning factor* $l \in [0, 1]$, which determines the sensitivity of G_1 to the change G_2 brings.

Focusing on the weighting function of the graph, resulting from the application of $U(G_1, G_2, l)$, the higher the value of learning factor, the higher the impact of the new graph to the resulting graph of the update. More precisely, a value of $l = 0$ indicates that G_1 will completely ignore the (considered new) graph G_2. A value of $l = 1$ indicates that the weights of the edges of G_1 will be

[2]In this case we chose non-overlapping n-grams, because n is minimal ($n = 2$). Nevertheless, we plan to evaluate whether this was a correct decision in the future.

assigned the values of the new graph's edges' weights. A value of 0.5 gives us the merging operator. The definition of the weighting performed in the graph resulting from U is:

$$W^i(e) = W^1(e) + (W^2(e) - W^1(e)) \times l \tag{2}$$

The U function allows using graphs to model a whole *set* of documents: in our case the model set. The model graph creation process comprises the initialization of a graph with the first document of the model set and the updating of that initial graph with the graphs of following model summaries. Especially, when one wants the overall graph's edges to hold weights averaging the weights of all the individual graphs that have contributed to it, then the i-th new graph that updates the overall graph should use a learning factor of $l = \frac{1}{i}, i > 1$. This gives a graph that has a role similar to the centroid of a set of vectors: it functions as a representative graph for the set its constituent graphs.

4. From Graph Matching to Summary Evaluation

Graph similarity calculation methods can be classified into two main categories.

Isomorphism-based: Isomorphism is a bijective mapping between the vertex set of two graphs V_1, V_2, such that all mapped vertices are equivalent, and every pair of vertices from V_1 shares the same state of neighborhood, as their corresponding vertices of V_2. In other words, in two isomorphic graphs all the nodes of one graph have their unique equivalent in the other graph, and the graphs also have identical connections between equivalent nodes. Based on the isomorphism, a *common subgraph* can be defined between V_1, V_2, as a subgraph of V_1 having an isomorphic equivalent graph V_3, which is a subgraph of V_2 as well. The *maximum common subgraph* of V_1 and V_2 is defined as the common subgraph with the maximum number of vertices. For more formal definitions and an excellent introduction to the error-tolerant graph matching, *i.e.,* fuzzy graph matching, see [Bun98].

Given the definition of the maximum common subgraph, a series of distance measures have been defined using various methods of calculation for the maximum common subgraph, or similar constructs like the Maximum Common Edge Subgraph, or Maximum Common Induced Graph (also see [RGW02]).

Edit-distance Based: Edit distance has been used in fuzzy string matching for some time now, using many variations (see [Nav01] for a survey on approximate string matching). The edit distance between two strings corresponds to the minimum number of edit character operations (namely insertion, deletion and replacement) needed to transform one string to the other. Based on this concept, a similar distance can be used for graphs [Bun98]. The edit operations for graphs' nodes are *node* deletion, insertion and substitution. The same three operations can by applied on edges, giving *edge* deletion, insertion and substitution.

Using a transformation from text to graph, the aforementioned graph matching methods can be used as a means to indicate text similarity.

We have applied the Value Similarity calculation [GKVS08], which offers graded similarity indication between two document graphs. Moreover, in order to compare a whole set of documents (model summaries) to a single evaluated text (evaluated summary) we represent the set of documents with a single graph, as we show in the following sections, whether be it an n-gram graph or a hierarchical proximity graph.

To compare two texts (or character sequences in general) T_1 and T_2 *e.g.,* for the task of summary evaluation against a gold standard text, we need to compare the texts' representations. Given that the representation of a text T_i is a set of graphs $\mathbb{G}_i$, containing graphs of various ranks, we use the *Value Similarity (VS)* for *every n-gram rank*, indicating how many of the edges contained in

graph G^i are contained in graph G^j, considering also the weights of the matching edges. In this measure each matching edge e having weight w_e^i in graph G^i contributes $\frac{\text{VR}(e)}{\max(|G^i|,|G^j|)}$ to the sum, while not matching edges do not contribute (consider that for an edge $e \notin G^i$ we define $w_e^i = 0$). The *ValueRatio (VR)* scaling factor is defined as:

$$\text{VR}(e) = \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)} \tag{3}$$

The equation indicates that the *ValueRatio* takes values in $[0, 1]$, and is symmetric. Thus, the full equation for *VS* is:

$$\text{VS}(G^i, G^j) = \frac{\sum_{e \in G^i} \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}}{\max(|G^i|, |G^j|)} \tag{4}$$

VS is a measure converging to 1 for graphs that share both the edges and similar weights, which means that a value of VS $= 1$ indicates perfect match between the compared graphs. Another important measure is the *Normalized Value Similarity (NVS)*, which is computed as:

$$\text{NVS}(G^i, G^j) = \frac{VS}{\frac{\min(|G^i|,|G^j|)}{\max(|G^i|,|G^j|)}} \tag{5}$$

The fraction $\text{SS}(G^i, G^j) = \frac{\min(|G^i|,|G^j|)}{\max(|G^i|,|G^j|)}$, is also called Size Similarity. The NVS is a measure of similarity where the ratio of sizes of the two compared graphs does not play a role. In the TAC case there is no real difference, however, because the SS factor is almost constant and equal to 1: the summaries have an almost fixed size. Thus, VS is equivalent to NVS.

The overall similarity VS^O of the sets $\mathbb{G}_1, \mathbb{G}_2$ is computed as the weighted sum of the VS over all ranks:

$$\text{VS}^O(\mathbb{G}_1, \mathbb{G}_2) = \frac{\sum_{r \in [L_{\min}, L_{\text{MAX}}]} r \times \text{VS}^r}{\sum_{r \in [L_{\min}, L_{\text{MAX}}]} r} \tag{6}$$

where VS^r is the VS measure for extracted graphs of rank r in $\mathbb{G}$, and $L_{\min}$, L_{MAX} are arbitrary chosen minimum and maximum n-gram ranks.

The similarity function calculation has a complexity of $O(|G_1| \times |G_2|)$, due to the fact that for each edge in G_1 one needs to lookup its identical edge in G_2. The similarity function calculation has a complexity of $O(|G_1| \times |G_2|)$, due to the fact that for each edge in G_1 one needs to lookup its identical edge in G_2. If an index is maintained with the edges' labels or the vertices' labels, this complexity can be diminished, which is the case in our implementation. Therefore, for every edge in the smallest of the two graphs, we perform a low complexity lookup in the edges of the biggest graph. If an edge is found we perform the calculation of the edge's contribution to the similarity sum. Otherwise, we continue with the next edge from the small graph. This gives a real complexity that is $O(h\min(|G_1|, |G_2|))$, where h is the constant time for a hash map lookup, if the edges are hashed. If the vertices are hashed, then the complexity is $O(h \min(|G_1|, |G_2|) \text{degree}(G_2))$, where the $\text{degree}(G_2)$ function returns the maximum number of edges connected to a single node in G_2.

For the comparison of HPGs we simply use, instead of ranks, the level of the graph in Equation 6. Thus, the similarity between two HPGs $\mathbb{J}_1$ and $\mathbb{J}_2$ is the weighted, normalized sum of value similarities between the corresponding levels of $\mathbb{J}_1$ and $\mathbb{J}_2$:

$$\text{VS}^O(\mathbb{J}_1, \mathbb{J}_2) = \frac{\sum_{l \in [1,L]} l \times \text{VS}^l}{\sum_{l \in [1,L]} r} \tag{7}$$

where VS^l is the VS measure for graphs of level l in $\mathbb{J}$.

In the AutoSummENG and the HPG cases, the grade of a summary is the average of the similarities to the model summaries — using the corresponding n-gram graph representations. In the case of MeMoG, there is only one similarity measurement, which is set to be the summary score. The performance of a summarization system is calculated as the average of its summary scores.

5. Experiments

The presented methods have been applied as part of the AESOP task of TAC2010. The task was "to create an automatic scoring metric for summaries, that would correlate highly with two manual methods of evaluating summaries, as applied in the TAC 2010 Guided Summarization task", namely the Pyramid method (modified pyramid score) [PMSG06] and Overall Responsiveness (see [DO08]).

The scoring metrics (better "measures") are to evaluate summaries including both model (*i.e.,* human generated) and automatic (non-model) summaries, produced within the TAC 2010 Guided Summarization task. In that task, 8 human summarizers produced a total of 368 model summaries, and 43 automatic summarizers produced a total of 3956 automatic summaries. The summaries are split into Main (or Initial) Summaries (Set A) and Update Summaries (Set B), according to the part of the Guided Summarization Task they fall into[3]. Two baseline summarizers were included in the set of automatic summarizers, namely one that uses the first sentences in the most recent document of a document set (ID:1). The second is the MEAD summarizer[4] (ID:2).

The experiments conducted upon the TAC 2010 corpus were based on the application of the AutoSummENG, the MeMoG , and the HPG methods on the TAC corpus. The n-gram graph parameters used for AutoSummENG were $(L_{\min}, L_{\text{MAX}}, D_{\text{win}}) = (3, 3, 3)$, which seems to offer near-optimal results on many English corpora. The same parameters were used for the MeMoG case. For the HPGs the distance factor parameter was set to 3, the minimum n-gram size to 2 and the number of levels $L = 5$.

We note that there were two different types of evaluation: the *All Peers* and the *No Models* evaluation. In the *No Models* case, the peer summaries are evaluated against all model summaries. In the *All Peers* case, the model summaries are evaluated against the remaining model summaries; on the other hand, the peer summaries are evaluated using jack-knifing against all the model summaries. The process of jack-knifing is the following. Given that there are 4 model summaries to evaluate against, the peer summary is evaluated against all combinations of the 4 models in groups of 3. The average grade assigned by all the evaluations is assigned to be the grade of the summary. The results of the evaluation, concerning the correlation to the Pyramid score are shown in Tables 1, 2; the results concerning the correlation to the Overall Responsiveness score are shown in Tables 3, 4. In the tables, the top performance for each column (*i.e.,* correlation test) is depicted in **bold**. All the correlation tests gave a p-value of less than 10^{-3}, making the results statistically highly significant.

We can easily determine that the MeMoG variant of the n-gram graphs application is the most promising variant when we aim to correlate best to the Pyramid score (Tables 1, 2). On the other hand, things are not as clear for the Overall Responsiveness case. There, the results illustrate that,

[3]See http://www.nist.gov/tac (Last visit: Feb 14, 2011) for more info on the Guided Summarization Task of TAC 2010.

[4]MEAD summarizer v3.12, publicly available at http://www.summarization.com/mead/ (Last visit: Feb 14, 2011). Default settings used, set to producing 100-word summaries.

Variant (ID)	Pearson (Rank)	Spearman (Rank)	Kendall (Rank)
Group A - All Peers			
AutoSummENG (25)	0.897 (10)	0.945 (8)	0.824 (7)
MeMoG (16)	**0.956 (3)**	**0.956 (4)**	**0.834 (4)**
HPG (12)	0.880 (15)	0.941 (10)	0.818 (9)
Group B - All Peers			
AutoSummENG (25)	0.807 (12)	0.920 (9)	0.779 (8)
MeMoG (16)	**0.968 (1)**	**0.935 (4)**	**0.799 (4)**
HPG (12)	0.726 (16)	0.931 (6)	0.788 (6)

TABLE 1. Correlation of grades to Pyramid score for All Peers

Variant (ID)	Pearson (Rank)	Spearman (Rank)	Kendall (Rank)
Group A - No Models			
AutoSummENG (25)	0.950 (11)	0.913 (9)	0.778 (10)
MeMoG (16)	**0.970 (4)**	**0.934 (5)**	**0.798 (6)**
HPG (12)	0.951 (10)	0.908 (10)	0.774 (11)
Group B - No Models			
AutoSummENG (25)	0.899 (13)	0.870 (11)	0.711 (11)
MeMoG (16)	**0.956 (5)**	**0.901 (7)**	**0.758 (6)**
HPG (12)	0.898 (14)	0.892 (9)	0.729 (9)

TABLE 2. Correlation of grades to Pyramid score without models

Variant (ID)	Pearson (Rank)	Spearman (Rank)	Kendall (Rank)
Group A - All Peers			
AutoSummENG (25)	0.909 (11)	0.942 (9)	**0.836 (4)**
MeMoG (16)	**0.944 (3)**	**0.949 (6)**	0.818 (10)
HPG (12)	0.892 (14)	0.945 (8)	0.820 (8)
Group B - All Peers			
AutoSummENG (25)	0.797 (11)	0.892 (9)	0.747 (9)
MeMoG (16)	**0.977 (1)**	0.915 (7)	0.772 (5)
HPG (12)	0.724 (15)	**0.925 (2)**	**0.777 (4)**

TABLE 3. Correlation of grades to Overall Responsiveness score for All Peers

even though MeMoG offers the best linear correlation to Responsiveness, it is the HPG that offers the best rank-based performance (Tables 3, 4).

Overall, the performance of both MeMoG and HPG in their corresponding domains is within the first 7 ranks between the 27 systems in the AESOP task. We should note that, in the top ranks, it is often the case that the difference in the performance of the top evaluator from the presented variants is very low (< 0.03). This means that the difference between performance may be attributed to chance, since the corresponding confidence intervals may well cover the 0.03 difference.

In Tables 5, 6 we illustrate the number of agreements and disagreements (see Appendix A for their definition) between measures, on whether the difference in performance between systems is statistically significant or not. Thus, the first row in Table 5 indicates that the verdict that came

Variant (ID)	Pearson (Rank)	Spearman (Rank)	Kendall (Rank)
Group A - No Models			
AutoSummENG (25)	0.923 (11)	0.911 (8)	**0.799 (5)**
MeMoG (16)	**0.949 (7)**	**0.920 (6)**	0.772 (8)
HPG (12)	0.932 (9)	0.916 (7)	0.779 (7)
Group B - No Models			
AutoSummENG (25)	0.876 (13)	0.829 (11)	0.686 (10)
MeMoG (16)	**0.933 (5)**	0.866 (9)	0.709 (8)
HPG (12)	0.888 (10)	**0.885 (5)**	**0.733 (6)**

TABLE 4. Correlation of grades to Overall Responsiveness score without models

Group A - All Peers

	Pyramid		Responsiveness	
	Agreement	Disagreement	Agreement	Disagreement
AutoSummENG (25)	280	64	280	64
MeMoG (16)	**344**	**0**	**344**	**0**
HPG (12)	200	144	200	144

Group B - All Peers

	Pyramid		Responsiveness	
	Agreement	Disagreement	Agreement	Disagreement
AutoSummENG (25)	284	60	284	60
MeMoG (16)	**344**	**0**	**344**	**0**
HPG (12)	157	187	157	187

TABLE 5. Agreement concerning discrimination - All Peers

by using AutoSummENG grades to determine statistically significant difference in performance between systems, agrees 813 times with the verdict when using Pyramid score, while it disagrees 90. There are also columns for the Responsiveness measure and there is differentiation between Group A (Main Summaries) and Group B (Update Summaries) texts.

We notice the excellent level of performance for the MeMoG variation in the All Peers case (marked in **bold**). It is also important to note that MeMoG is the only variation the grades of which are affected by jack-knifing (due to its non-linearity).

6. Conclusion - Future work

This paper briefly described two novel methods for summarization system evaluation. The first, named MeMoG, relies on a merged n-gram graph representation for the model summaries. The second, named HPG, relies on a hierarchy of proximity graphs — which are a generalization of n-gram graphs — to represent model texts. Both methods were applied on the TAC 2010 AESOP task corpus, offering very promising results in different aspects of the evaluation. Both methods offered an improvement over the AutoSummENG method, pointing into new directions for statistical summary evaluation.

In the future, we plan to optimize parameters for the HPG method and apply all the n-gram graph based methods to corpora of varying languages to support our language-neutrality claim.

Group A - No Models				
	Pyramid		Responsiveness	
	Agreement	Disagreement	Agreement	Disagreement
AutoSummENG (25)	813	90	793	110
MeMoG (16)	798	105	788	115
HPG (12)	798	105	788	115
Group B - No Models				
	Pyramid		Responsiveness	
	Agreement	Disagreement	Agreement	Disagreement
AutoSummENG (25)	742	161	723	180
MeMoG (16)	764	139	753	150
HPG (12)	792	111	787	114

TABLE 6. Agreement concerning discrimination - No Models

ACKNOWLEDGEMENTS

This work was completed while George Giannakopoulos was working for the University of Trento, Italy.

REFERENCES

[Bun98] H. Bunke. Error-tolerant graph matching: a formal framework and algorithms. *Advances in Pattern Recognition, LNCS*, 1451:1–14, 1998.

[BV04] Michele Banko and Lucy Vanderwende. Using n-grams to understand the nature of summaries. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 1–4, Boston, Massachusetts, USA, May 2004. Association for Computational Linguistics.

[CS04] T. Copeck and S. Szpakowicz. Vocabulary usage in newswire summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 19–26. Association for Computational Linguistics, 2004.

[Dan05] H. T. Dang. Overview of DUC 2005. In *Proceedings of the Document Understanding Conf. Wksp. 2005 (DUC 2005) at the Human Language Technology Conf./Conf. on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005)*, 2005.

[DO08] H. T. Dang and K. Owczarzak. Overview of the TAC 2008 update summarization task. In *TAC 2008 Workshop - Notebook papers and results*, pages 10–23, Maryland MD, USA, November 2008.

[GKVS08] George Giannakopoulos, Vangelis Karkaletsis, George Vouros, and Panagiotis Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *ACM Trans. Speech Lang. Process.*, 5(3):1–39, 2008.

[HLZ05] E. Hovy, C. Y. Lin, and L. Zhou. Evaluating duc 2005 using basic elements. *Proceedings of DUC-2005*, 2005.

[HLZF05] E. Hovy, C. Y. Lin, L. Zhou, and J. Fukumoto. Basic elements, 2005.

[LH03] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 71–78, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Lin04] C. Y. Lin. Rouge: A package for automatic evaluation of summaries. *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26, 2004.

[Nav01] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[PMSG06] R. J. Passonneau, K. McKeown, S. Sigelman, and A. Goodkind. Applying the pyramid method in the 2006 document understanding conference. In *Proceedings of Document Understanding Conference (DUC) Workshop 2006*, 2006.

[RGW02] J. W. Raymond, E. J. Gardiner, and P. Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631, 2002.

[SK09] F. Schilder and R. Kondadadi. *A Metric for Automatically Evaluating Coherent Summaries via Context Chains*, pages 65–70. 2009.

[ZLMH06] L. Zhou, C. Y. Lin, D. S. Munteanu, and E. Hovy. Paraeval: Using paraphrases to evaluate summaries automatically. In *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL-2006)*, 2006.

Appendix A. Discriminative Power Definitions

The following example and definitions originate from the README_aesop.txt file in the evaluation results, kindly provided by NIST.

```
                  Pyramid
Submission 7
  224 294   0
  143 820   0
    0   0   4

(Column1,Row1) shows the number of pairs of summarizers (X,Y), where
the AESOP metric (Submission 7) and the Pyramid method agree that
summarizer X is significantly better than summarizer Y, or that
summarizer Y is significantly better than summarizer X (here: 224
agreements).

(Column2,Row1) shows the number of pairs of summarizers (X,Y), where
there is a significant difference between X and Y according to the
AESOP metric, but there is no significant difference according to the
Pyramid method (here: 294 disagreements).

(Column1,Row2) shows the number of pairs of summarizers (X,Y), where
there is a significant difference between X and Y according to the
Pyramid method, but there is no significant difference according to
the AESOP metric (here: 143 disagreements).

(Column2,Row2) shows the number of pairs of summarizers (X,Y), where
the AESOP metric and Pyramid method agree that there is no significant
difference between summarizer X and summarizer Y (here: 820
agreements).

(Column3,Row3) shows the number of pairs of summarizers (X,Y), where
the AESOP metric and the Pyramid method both say that summarizer X is
significantly different from summarizer Y, but disagree as to which
summarizer is better (here: 4 disagreements). Either

    (1) according to the AESOP metric X < Y,
    and according to Pyramid Y < X; or

    (2) according to the AESOP metric Y < X,
    and according to Pyramid X < Y.

The remaining cells can be ignored.
```

In this work, we consider as *Agreements* the sum of cells $(Column1, Row1) + (Column2, Row2)$ and as *Disagreements* the sum of the remaining cells $(Column1, Row2) + (Column2, Row1) + (Column3, Row3)$. The data were provided by NIST.