A Framework for Identifying Textual Redundancy

Kapil Thadani and **Kathleen McKeown**
Department of Computer Science,
Columbia University,
New York, NY USA
{kapil,kathy}@cs.columbia.edu

Abstract

The task of identifying redundant information in documents that are generated from multiple sources provides a significant challenge for summarization and QA systems. Traditional clustering techniques detect redundancy at the sentential level and do not guarantee the preservation of all information within the document. We discuss an algorithm that generates a novel graph-based representation for a document and then utilizes a set cover approximation algorithm to remove redundant text from it. Our experiments show that this approach offers a significant performance advantage over clustering when evaluated over an annotated dataset.

1 Introduction

This paper approaches the problem of identifying and reducing redundant information in documents that are generated from multiple sources. This task is closely related to many well-studied problems in the field of natural language processing such as summarization and paraphrase recognition. Systems that utilize data from multiple sources, such as question-answering and extractive summarization systems that operate on news data, usually include a component to remove redundant information from appearing in their generated output.

However, practical attempts at reducing redundancy in the output of these types of systems usually involve clustering the sentences of the generated output, picking a representative sentence from each cluster and discarding the rest. Although this strategy would remove some redundant information, clustering approaches tuned for coarse

matches could also remove non-redundant information whereas clustering approaches tuned for near-exact matches could end up removing very little repeated information. This is simply a consequence of the fact that information can, and usually does, exist at the sub-sentential level and that clusters of sentences don't necessarily correspond to clusters of *information*.

In this paper, we discuss a framework for building a novel graph-based representation to detect redundancy within documents. We identify redundancy at the sub-sentential level through pairwise alignment between the sentences of a document and use this to build a bipartite graph which enables us to keep track of redundant information across all sentences. Common information between pairs of sentences, detected with the alignment algorithm, can be extrapolated to document-wide units of information using the graph structure. Individual sentences that are encompassed by the information in the rest of the document can then be identified and removed efficiently by using a well-known greedy algorithm adapted for this representation.

2 Related Work

The challenge of minimizing redundant information is commonly faced by IR engines and extractive summarization systems when generating their responses. A well-known diversity-based reranking technique for these types of systems is MMR (Carbonell and Goldstein, 1998), which attempts to reduce redundancy by preferring sentences that differ from the sentences already selected for the summary. However, this approach does not attempt to identify sub-sentential redundancy.

Alternative approaches to identifying redundancy use clustering at the sentence level (Lin and Hovy, 2001) to remove sentences that are largely repetitive; however, as noted earlier, this is not well-suited to the redundancy task. The use of sen-

tence simplification in conjunction with clustering (Siddharthan et al., 2004) could help alleviate this problem by effectively clustering smaller units, but this issue cannot be avoided unless sentences are simplified to atomic elements of information.

Other research has introduced the notion of identifying concepts in the input text (Filatova and Hatzivassiloglou, 2004), using a *set cover* algorithm to attempt to include as many concepts as possible. However, this approach uses *tf-idf* to approximate concepts and thus doesn't explicitly identify redundant text. Our work draws on this approach but extends it to identify all detectable redundancies within a document set.

Another approach does identify small subsentential units of information within text called "Basic Elements" and uses these for evaluating summarizations (Hovy et al., 2006). Our approach, in contrast, does not make assumptions about the size or structure of redundant information since this is uncovered through alignments.

We thus require the use of an alignment algorithm to extract the common information between two pieces of text. This is related to the well-studied problem of identifying paraphrases (Barzilay and Lee, 2003; Pang et al., 2003) and the more general variant of recognizing textual entailment, which explores whether information expressed in a hypothesis can be inferred from a given premise. Entailment problems have also been approached with a wide variety of techniques, one of which is dependency tree alignment (Marsi et al., 2006), which we utilize as well to align segments of text while respecting syntax. However, our definition of redundancy does not extend to include unidirectional entailment, and the alignment process is simply required to identify equivalent information.

3 Levels of Information

In describing the redundancy task, we deal with multiple levels of semantic abstraction from the basic lexical form. This section describes the terminology used in this paper and the graph-based representation that is central to our approach.

3.1 Terminology

The following terms are used throughout this paper to refer to different aspects of a document.

Snippet: This is any span of text in the document and is a lexical realization of information. While a snippet generally refers to a single sentence within a document, it can apply to multiple sentences or phrases within sentences. Since redundancy will be reduced by removing whole snippets, a snippet can be defined as the smallest unit of text that can be dropped from a document for the purpose of reducing redundancy.

To illustrate the levels of information that we consider, consider the following set of short sentences as snippets. Although this is a synthesized example to simplify presentation, sentences with this type of overlapping information occur frequently in the question-answering scenario over news in which our approach has been used.

$\mathbf{x}_1$: Whittington is an attorney.

$\mathbf{x}_2$: Cheney shot Whittington, a lawyer.

$\mathbf{x}_3$: Whittington, an attorney, was shot in Texas.

$\mathbf{x}_4$: Whittington was shot by Cheney while hunting quail.

$\mathbf{x}_5$: This happened during a quail hunt in Texas.

We can see that all the information in $\mathbf{x}_1$ is contained in both $\mathbf{x}_2$ and $\mathbf{x}_3$. While no other snippet is completely subsumed by any single snippet, they can be made redundant given combinations of other snippets; for example, $\mathbf{x}_4$ is redundant given $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_5$. In order to identify these combinations, we need to identify the elements of information within each snippet.

Concept: This refers to a basic unit of information within a document. Concepts may be facts, opinions or details. These are not necessarily *sememes*, which are atomic units of meaning, but simply units of information that are seen as atomic within the document. We further restrict the definition of a concept to a unit of information seen in *more than one snippet*, since we are only interested in concepts which help in identifying redundancy.

Formally, a document can be defined as a set of S snippets $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$, which is a literal representation of the document. However, it can also be defined in terms of its *information content*. We use $\mathbf{Z} = \{\mathbf{z}^1, \ldots, \mathbf{z}^C\}$ to represent the set of C concepts that cover all information appearing more than once in the document. In the example above, we can identify five non-overlapping concepts:

$\mathbf{z}^{\mathrm{A}}$: Whittington was shot

$\mathbf{z}^{\mathrm{B}}$: Whittington is an attorney

$\mathbf{z}^{\mathrm{C}}$: The shooting occurred in Texas

$\mathbf{z}^{\mathrm{D}}$: It happened during a hunt for quail

$\mathbf{z}^{\mathrm{E}}$: Cheney was the shooter

We use subscripts for snippet indices and superscripts for concept indices throughout this paper.

Nugget: This is a textual representation of a concept in a snippet and therefore expresses some information which is also expressed elsewhere in the document. Different nuggets for a given concept may have unique lexico-syntactic realizations, as long as they all embody the same semantics. With regard to the notation used above, nuggets can be represented by an $S \times C$ matrix $\mathbf{Y}$ where each $\mathbf{y}_s^c$ denotes the fragment of text (if any) from the snippet $\mathbf{x}_s$ that represents concept $\mathbf{z}^c$.

Since a concept itself has no unique textual realization, it can be simply represented by the combination of all its nuggets. For instance, in the example shown above, concept $\mathbf{z}^D$ is seen in both $\mathbf{x}_4$ and $\mathbf{x}_5$ in the form of two nuggets $\mathbf{y}_4^D$ ("... while hunting quail") and $\mathbf{y}_5^D$ ("... during a quail hunt"), which are paraphrases. The degree to which we can consider this and other types of lexical or syntactic differences between nuggets that have the same semantic identity depends on the alignment algorithm used.

Intersection: This is the common information between two snippets that can be obtained through their alignment. For example, the intersection from the alignment between $\mathbf{x}_2$ and $\mathbf{x}_4$ consists of two fragments of text that express that Cheney shot Whittington (an active-voiced fragment from $\mathbf{x}_2$ and a passive-voiced fragment from $\mathbf{x}_4$).

In general, aligning $\mathbf{x}_i$ and $\mathbf{x}_j$ produces an intersection $\mathbf{v}_{i,j}$ which is simply a pair of aligned text fragments covering the set of concepts that $\mathbf{x}_i$ and $\mathbf{x}_j$ have in common. However, these undivided segments of text may actually contain multiple nuggets from a document-wide perspective. We assume that intersections can be *decomposed* into smaller intersections through further alignments with snippets or other intersections; this process is explained in Subsection 4.3.

3.2 Concept graph representation

Figure 1 illustrates the example introduced in Subsection 3.1 as a network with intersections represented as edges between snippets. This is the type of graph that would be built using pairwise alignments between all snippets. Note that although some intersections such as $\mathbf{v}_{1,2}$ (between $\mathbf{x}_1$ and $\mathbf{x}_2$) and $\mathbf{v}_{3,5}$ express concepts directly, other intersections such as $\mathbf{v}_{2,3}$ and $\mathbf{v}_{2,4}$ are undivided combinations of concepts. Since we cannot directly identify concepts and their nuggets, this graph is not immediately useful for reducing redundancy.
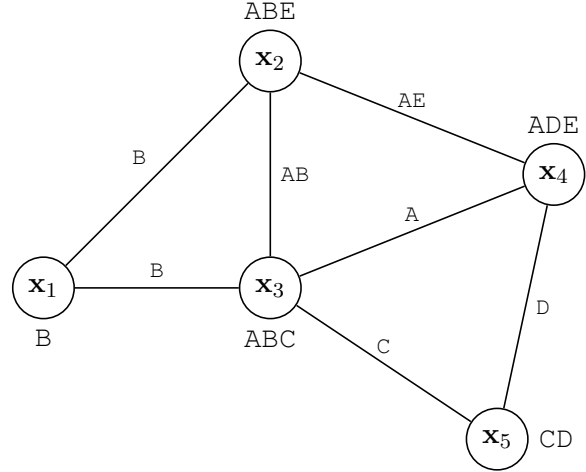

Figure 1: Graph representing pairwise alignments between the example snippets from Section 3. For clarity, alphabetic labels like A represent concepts $\mathbf{z}^A$. Node labels show concepts within snippets and edge labels indicate concepts seen in intersections.
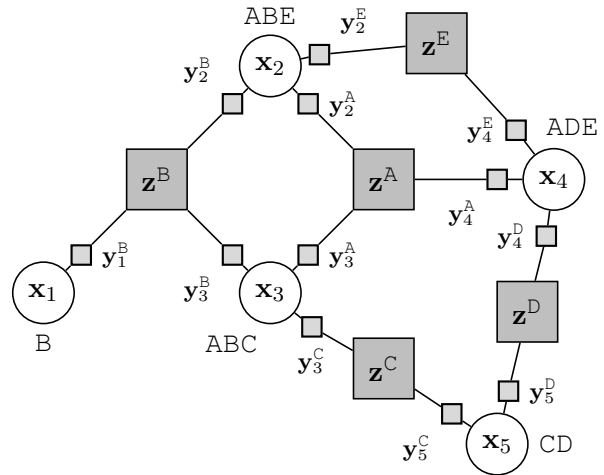

Figure 2: Structure of the equivalent concept graph for the example document illustrated in Figure 1. Circular nodes $\mathbf{x}_s$ represent snippets, large squares $\mathbf{z}^c$ represent concepts and small squares $\mathbf{y}_s^c$ depict nuggets for each concept within a snippet.

Now, since the matrix $\mathbf{Y}$ describes the interaction of concepts with snippets, it can be viewed as an incidence matrix that defines a bipartite graph between snippets and concepts with nuggets representing the edges. In this *concept graph* representation, each snippet can connect to any number of other snippets via a shared concept. Since concepts serve to connect multiple snippets together, the concept graph can also be seen as a hypergraph, which is a generalization of a graph in which each edge may connect together multiple vertices.

Figure 2 illustrates the structure of the equivalent concept graph for the previous example. This is simply the bipartite graph with the two types of nodes, namely snippets and concepts, represented using different symbols. For clarity, nuggets are also depicted as nodes in the graph, thereby reducing edges to simple links indicating membership. This representation identifies the redundancy between snippets in terms of non-overlapping concepts and is therefore more useful than the graph from Figure 1 for reducing redundancy.

4 Constructing the Concept Graph

We now describe how a concept graph can be constructed from a document by using dependency tree alignment and leveraging the existing structure of the graph during construction.

4.1 Alignment of snippets

In order to obtain the concept graph representation, the common information between each pair of snippets in the document must first be discovered by aligning all pairs of snippets with each other. We make use of dependency parsing and alignment of dependency parse trees to obtain intersections between each pair of snippets, where each intersection may be a discontiguous span of text corresponding to an aligned subtree within each snippet. In our experiments, dependency parsing is accomplished with Minipar (Lin, 1998) and alignment is done using a bottom-up tree alignment algorithm (Barzilay and McKeown, 2005) modified to account for the shallow semantic role labels produced by the parser. The alignment implementation is not the focus of this work, however, and the framework described here could by applied using any alignment technique between segments of text in potentially any language.

As seen in Figure 1, the intersections that can be extracted solely by pairwise comparisons are not unique and may contain multiple concepts. A truly information-preserving approach requires the explicit identification of concepts as in the concept graph from Figure 2, but efficiently converting the former into the latter poses a non-trivial challenge.

4.2 Extraction of irreducible concepts

Our approach attempts to obtain a set of *irreducible concepts* such that each concept in this set cannot wholly or partially contain any other concept in the set (thereby conforming to the definition of a concept in Subsection 3.1).

We attempt to build the concept graph and maintain irreducible concepts alongside each of the $S(S-1)/2$ pairwise alignment steps. Every intersection found by aligning a pair of snippets is assumed to represent some concept that these snippets share; it is then compared with existing concepts and is decomposed into smaller intersections if it overlaps partially with any one of them. This implies a worst-case of C comparisons at each pairwise alignment step ($2C$ if both fragments of an intersection are compared separately). However, this can be made more efficient by exploiting the structure of the graph. A new intersection only has to be compared with concepts which might be affected by it and only affects the other snippets containing these concepts. We can show that this leads to an algorithm that requires fewer than C comparisons, and additionally, that these comparisons can be performed efficiently.

Consider the definition of alignment along the lines of a mathematical relation. We require snippet alignment to be an *equivalence relation* and it therefore must have the following properties.

Symmetry: If an intersection $\mathbf{v}_{i,j}$ contains a concept $\mathbf{z}'$, then $\mathbf{v}_{j,i}$ will also contain $\mathbf{z}'$. This property allows only $S(S-1)/2$ alignments to suffice instead of the full $S(S-1)$. Therefore, without loss of generality, we can specify that all alignments between $\mathbf{x}_i$ and $\mathbf{x}_j$ should have $i < j$.

Transitivity: If intersections $\mathbf{v}_{i,j}$ and $\mathbf{v}_{j,k}$ both contain some concept $\mathbf{z}'$, then $\mathbf{v}_{i,k}$ will also contain $\mathbf{z}'$. This property leads to an interesting consequence. Assuming we perform alignments in order (initially aligning $\mathbf{x}_1$ and $\mathbf{x}_2$ and iterating for j within each i), we observe that $\mathbf{x}_i$ has been aligned with snippets $\{\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}\}$ and, for any $i > 1$, snippets $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}\}$ were aligned with all snippets $\{\mathbf{x}_1, \ldots, \mathbf{x}_S\}$. Since $i < j$, this implies that $\mathbf{x}_i$ was directly aligned with snippets $\{\mathbf{x}_1, \ldots, \mathbf{x}_{i-1}\}$ which in turn were each aligned with all S snippets. Therefore, due to the property of transitivity, all concepts contained in a new intersection $\mathbf{v}_{i,j}$ that *also* exist in the partly-constructed graph would already be directly associated with $\mathbf{x}_i$. Note that this does not hold for $\mathbf{x}_j$ as well, since $\mathbf{x}_j$ has not been aligned with $\{\mathbf{x}_{i+1}, \ldots, \mathbf{x}_{j-1}\}$; therefore, it may not have encountered all relevant concepts.

This implies that for any i and j, all concepts that might be affected by a new intersection $\mathbf{v}_{i,j}$

have already been uncovered in $\mathbf{x}_i$ and thus $\mathbf{v}_{i,j}$ only needs to be compared to these concepts.

4.3 Comparisons after alignment

For every new intersection $\mathbf{v}_{i,j}$ produced by an alignment between $\mathbf{x}_i$ and $\mathbf{x}_j$, the algorithm compares it (specifically, the fragment from $\mathbf{x}_i$) with each existing nugget $\mathbf{y}_i^k$ for each concept $\mathbf{z}^k$ already seen in $\mathbf{x}_i$. Checking for the following cases ensures that the graph structure contains only irreducible concepts for all the alignments seen:

1. If $\mathbf{v}_{i,j}$ doesn't overlap with any current nugget from $\mathbf{x}_i$, it becomes a new concept that links to $\mathbf{x}_i$ and $\mathbf{x}_j$. In our example, the first intersection $\mathbf{v}_{1,2}$ contains "Whittington ... an attorney" from $\mathbf{x}_1$ and "... Whittington, a lawyer" from $\mathbf{x}_2$; this becomes a new concept $\mathbf{z}^\mathrm{B}$ since $\mathbf{x}_1$ has no other nuggets.

2. If $\mathbf{v}_{i,j}$ overlaps completely with a nugget $\mathbf{y}_i^k$, then $\mathbf{x}_j$ must also be linked to concept $\mathbf{z}^k$. For example, $\mathbf{x}_1$'s fragment in the second intersection $\mathbf{v}_{1,3}$ is also "Whittington ... an attorney", so $\mathbf{x}_3$ must also link to $\mathbf{z}^\mathrm{B}$.

3. If $\mathbf{v}_{i,j}$ subsumes $\mathbf{y}_i^k$, it is split up and the non-overlapping portion is rechecked against existing nuggets recursively. For example, $\mathbf{x}_2$'s fragment in the intersection $\mathbf{v}_{2,3}$ is "... shot Whittington, a lawyer", part of which overlaps with $\mathbf{y}_2^B$, so this intersection is divided up and the part representing "... shot Whittington ..." becomes a new concept $\mathbf{z}^A$.

4. If, on the other hand, $\mathbf{y}_i^k$ subsumes $\mathbf{v}_{i,j}$, the concept $\mathbf{z}^k$ is itself split up along with all nuggets that it links to, utilizing the present structure of the graph.

When comparing intersections, we can restrict the decomposition of nuggets to prevent the creation of overly-granular concepts. For instance, we can filter out intersections containing only isolated named-entities or syntactic artifacts like determiners since they contain no information by themselves. We can also prevent verbs and their arguments from being split apart using information from a snippet's dependency parse, if available.

4.4 Efficiency of the algorithm

Instead of C additional comparisons in the worst case after each pairwise snippet alignment, we need no more comparisons in the worst case than the maximum number of concepts that can exist in a single snippet. Since this value grows no faster than C as S increases, this is a significant improvement. Other factors, such as the overhead required to split up concepts, remain unchanged.

Furthermore, since all the additional comparisons are carried out between nuggets of the *same* snippet, we don't need to perform any further alignment among nuggets or concepts. Alignments are expensive; each is $O(n_1 n_2)$ where n_1 and n_2 are the number of words in the two segments of text being aligned (if dependency tree alignment is used) along with an overhead for checking word similarity. However, since we now only need to compare text from the same snippet, the comparison can be performed in linear time by simply comparing spans of word indices, thereby also eliminating the overhead for comparing words.

5 Decreasing redundancy

The concept graph can now be applied to the task of reducing redundancy in the document by dropping snippets which contain no information that is not already present in the rest of the document.

5.1 Reduction to set cover

Every snippet $\mathbf{x}_s$ in a document can be represented as a set of concepts $\{\mathbf{z}^c : \mathbf{y}_s^c \in \mathbf{Y}\}$. Since concepts are defined as information that is seen in more than one snippet as per the definition in Subsection 3.1, representing snippets as sets of concepts will overlook any unique information present in a snippet. Without loss of generality, we can add any such unique information in the form of an *artificial concept* for each snippet to $\mathbf{Z}$ so that snippets can be completely represented as sets of concepts from $\mathbf{Z}$. Note that the union of snippets $\bigcup_{s=1}^{S} \mathbf{x}_s$ equals $\mathbf{Z}$.

Reducing redundancy in the document while preserving all information requires us to identify the *most* snippets whose entire informational content is covered by the rest of the snippets in the document, thereby targeting them for removal. Since we express informational content in concepts, this problem reduces to the task of finding the smallest group of snippets that together cover all the concepts in the document, i.e. we need to find the *smallest* subset $\mathbf{X}' \subseteq \mathbf{X}$ such that, if $\mathbf{X}'$ contains R snippets $\mathbf{x}'_r$, the union of these snippets $\bigcup_{r=1}^{R} \mathbf{x}'_r$ also equals $\mathbf{Z}$. Therefore, every concept in a snippet from $\mathbf{X} - \mathbf{X}'$ also exists in at least one snippet from $\mathbf{X}'$ and no concept from $\mathbf{Z}$ is lost.

This formulation of the problem is the classic

set cover problem, which seeks to find the smallest possible group of subsets of a universe that covers all the other subsets. A more general variant of this problem is *weighted set cover* in which the subsets have weights to be maximized or costs to be minimized. While this problem is known to be NP-hard, there exists a straightforward local maximization approach (Hochbaum, 1997) which runs in polynomial time and is proven to give solutions within a known bound of the optimal solution. This greedy approximation algorithm can be adapted to our representation.

5.2 Selecting non-redundant snippets

The algorithm selects a snippet $\mathbf{x}_r$ to the subset $\mathbf{X}'$ such that *information content* of $\mathbf{X} \cup \mathbf{x}_r$ is maximized. In general, this implies that the snippet with the highest degree over uncovered concepts must be selected at each iteration. Other measures such as snippet length, fluency, or rank in an ordered list can be included in a weight measure in order to break ties and introduce a preference for shorter, more fluent, or higher-ranked snippets.

Consider the example from Section 3. The candidates for selection are $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$ since they contain the most *uncovered* concepts. If $\mathbf{x}_2$ is selected, its concepts $\mathbf{z}^A$, $\mathbf{z}^B$ and $\mathbf{z}^E$ are *covered*. At this stage, $\mathbf{x}_5$ contains two uncovered concepts while $\mathbf{x}_3$ and $\mathbf{x}_4$ contain just one each. Thus, $\mathbf{x}_5$ is selected next and its concepts $\mathbf{z}^C$ and $\mathbf{z}^D$ are covered. Since no uncovered concepts remain, all snippets which haven't been selected are redundant. This solution, which is shown in Figure 3, selects the following text to cover all the snippets:

$\mathbf{x}_2$: Cheney shot Whittington, a lawyer.

$\mathbf{x}_5$: This happened during a quail hunt in Texas.

Other solutions are also possible depending on the factors involved in choosing the snippet to be selected at each iteration. For example, the algorithm might choose to select $\mathbf{x}_3$ first instead of $\mathbf{x}_2$, thereby yielding the following solution:

$\mathbf{x}_3$: Whittington, an attorney, was shot in Texas.

$\mathbf{x}_4$: Whittington was shot by Cheney while hunting quail.

6 Experiments

To evaluate the effectiveness of this framework empirically, we ran experiments over documents containing annotations corresponding to concepts within the document. We also defined a metric
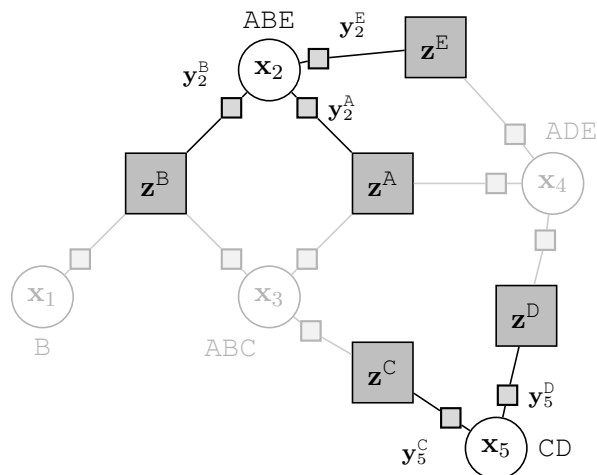

Figure 3: Pruned version of the concept graph example shown in Figure 2, illustrating the outcome of removing redundant snippets.

for comparing any concept graph over a document to a gold-standard concept graph. This was used to compare the concept graphs created by our approach to perturbed versions of the gold-standard graphs and graphs created by clustering.

6.1 Dataset

Due to the scarcity of available annotated datasets suitable for evaluating redundancy, we utilized the pyramid dataset from DUC 2005 (Nenkova et al., 2007) which was created from 20 articles for the purpose of summarization evaluation. Each pyramid document is a hierarchical representation of 7 summaries of the orginal news article. These summaries have been annotated to identify the individual *semantic content units* or SCUs where each SCU represents a certain fact, observation or piece of information in the summary. A sentence fragment representing an occurrence of an SCU in a summary is a *contributor* to the SCU.

The pyramid construction for a group of summaries of the same article mirrors the concept graph representation described in Subsection 3.2. SCUs with more than two contributors are similar in definition to concepts while their contributors fill the role of nuggets. Using this analogy, each dataset consists of a combination of the seven summaries in a single pyramid document; the 20 pyramid documents therefore yield 20 datasets.

6.2 Evaluation metrics

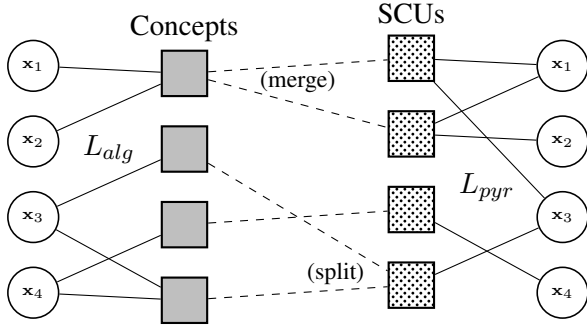The evaluation task requires us to compare the concept graph generated by our algorithm to the ideal

Figure 4: The bipartite graph on the left shows snippets $\mathbf{x}_s$ linked to concepts produced automatically; the one on the right shows the same snippets linked to SCUs from annotated data. Dashed lines indicate mappings between concepts and SCUs.

Measure	Random	Clustering	Concepts
Precision	0.0510	0.2961	**0.4496**
Recall	0.0515	0.1162	**0.3266**
F_1 score	0.0512	0.1669	**0.3783**

Table 1: Summary of the evaluation metrics averaged over all 20 pyramid documents when $m = 0.5$

concept graph extracted from the pyramid document annotations. Standard metrics do not apply easily to the problem of comparing bipartite graphs, so we define a novel metric modeled on the well-known IR measures of precision, recall and F-measure. Figure 4 illustrates the elements involved in the evaluation task.

We define the metrics of precision, recall and F-measure over the *links* between snippets and concepts. Assuming we have a mapping between generated concepts and gold-standard SCUs, we can judge whether each link is correct. Let each single link between a snippet and a concept have an associated weight of 1 by default and let L indicate a set of such links. We use L_{alg} and L_{pyr} to distinguish between the sets of links generated by the algorithm and retrieved from the annotations respectively. Precision and recall are defined as follows while F-measure retains its traditional definition as their harmonic mean.

$$\text{Precision} = \frac{\text{Sum of weights in } L_{alg} \cap L_{pyr}}{\text{Sum of weights in } L_{alg}}$$

$$\text{Recall} = \frac{\text{Sum of weights in } L_{alg} \cap L_{pyr}}{\text{Sum of weights in } L_{pyr}}$$

To determine a mapping between concepts and SCUs, we identify every concept and SCU pair, say $\mathbf{z}^c$ and $\mathbf{z}^s$, which has one or more snippets in common and, for each snippet $\mathbf{x}_i$ that they have in common, we find the longest common subsequence between their nuggets $\mathbf{y}_i^c$ and $\mathbf{y}_i^s$ to obtain the following score which ranges from 0 to 1.

$$\text{LCS score} = \frac{length(\text{LCS})}{\min\left(length(\mathbf{y}_i^c), length(\mathbf{y}_i^s)\right)}$$

This score is compared with a user-defined mapping threshold m to determine if the concept and SCU are sufficiently similar. In order to avoid biasing the metric by permitting multiple mappings per concept, we adjust for *merges* or $1 : N$ mappings by cloning the concept and creating N $1 : 1$ mappings in its place. We then adjust for *splits* or $N : 1$ mappings by dividing the weight of each of the links connected to a participating concept by N. Due to this normalization, the metrics are observed to be stable over variations in m.

6.3 Baselines

We compare the performance of the algorithm against two baselines. The first approach involves a random concept assignment scheme to build artificial concept graphs using the distributional properties of the gold-standard concept graphs. The number of concepts C and the number of snippets that each concept links to is determined by sampling from distributions over these properties derived from the statistics of the actual SCU graph for that document. For evaluation, these artificial concepts are randomly mapped to SCUs using m to control the likelihood of mapping. The best scores from 100 evaluation runs were considered.

The second baseline used for comparison is a clustering algorithm, since clustering is the most common approach to dealing with redundancy. For this purpose, we use a recursive spectral partitioning algorithm, a variant of spectral clustering (Shi and Malik, 2000) which obtains an average V-measure (Rosenberg and Hirschberg, 2007) of 0.93 when clustering just pyramid contributors labeled by their SCUs. The algorithm requires a parameter that controls the homogeneity of each cluster; we run it over the entire range of settings of this parameter. We consider the clustering that *maximizes* F-measure in order to avoid any uncertainty regarding optimal parameter selection and to implicitly compare our algorithm against an entire hierarchy of possible clusterings.

6.4 Results

Table 1 shows the F_1 scores over evaluation runs using the random concept assignment, clustering and concept graph techniques. These results are obtained at a mapping threshold of $m = 0.5$, which implies that we consider a mapping between a concept and an SCU if their nuggets over common sentences share more than 50% of their words on average. The results do not vary significantly at different settings of m.

We observe that the concepts extracted by our graph-based approach perform significantly better than the best-performing clustering configuration. Despite a fairly limited alignment approach that doesn't use synonyms or semantic analysis, the concept graph outperforms the baselines by nearly an order of magnitude on each document. This validates our initial hypothesis that clustering approaches are not suitable for tackling the redundancy problem at the sub-sentential level.

7 Conclusions and Future Work

We have described a graph-based algorithm for identifying redundancy at the sub-snippet level and shown that it outperforms clustering methods that are traditionally applied to the redundancy task.

Though the algorithm identifies redundancy at the sub-snippet level, redundancy can be decreased by dropping only entirely redundant snippets. We hope to be able to overcome this limitation by extending this information-preserving approach to the synthesis of new non-redundant snippets which minimize redundant content in the document.

In addition, this work currently assumes that redundancy is bidirectional; however, we intend to also address the case of unidirectional redundancy by considering entailment recognition approaches.

Acknowledgements

References

Barzilay, Regina and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of HLT-NAACL*, pages 16–23.

Barzilay, Regina and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Carbonell, Jaime G. and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of ACM-SIGIR*, pages 335–336.

Filatova, Elena and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence text extraction. In *Proceedings of COLING*, page 397.

Hochbaum, Dorit S. 1997. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In *Approximation algorithms for NP-hard problems*, pages 94–143. PWS Publishing Co., Boston, MA, USA.

Hovy, Eduard, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of LREC*.

Lin, Chin-Yew and Eduard Hovy. 2001. From single to multi-document summarization: a prototype system and its evaluation. In *Proceedings of ACL*, pages 457–464.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, LREC*.

Marsi, Erwin, Emiel Krahmer, Wauter Bosma, and Mariet Theune. 2006. Normalized alignment of dependency trees for detecting textual entailment. In *Second PASCAL Recognising Textual Entailment Challenge*, pages 56–61.

Nenkova, Ani, Rebecca Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing*, 4(2):4.

Pang, Bo, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*, pages 102–109.

Rosenberg, Andrew and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP*, pages 410–420.

Shi, Jianbo and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Siddharthan, Advaith, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of COLING*, page 896.