Automated Pyramid Summarization Evaluation

Yanjun Gao[1], **Chen Sun**[2], and **Rebecca J. Passonneau**[1]

[1,2]Department of Computer Science and Engineering
Pennsylvania State University
[1] {yug125,rjp49}@cse.psu.edu
[2]chensunx@gmail.com

Abstract

Pyramid evaluation was developed to assess the content of paragraph length summaries of source texts. A pyramid lists the distinct units of content found in several reference summaries, weights content units by how many reference summaries they occur in, and produces three scores based on the weighted content of new summaries. We present an automated method that is more efficient, more transparent, and more complete than previous automated pyramid methods. It is tested on a new dataset of student summaries, and historical NIST data from extractive summarizers.

1 Introduction

During the 70's and 80's, educational pychologists studied human summarization skills, and their development throughout secondary school and beyond. Three separate skills are acquired in the following order: selection of **important** information, **abstraction** through vocabulary generalization and sentence fusion, and **integration** with background knowledge (van Dijk and Kintsch, 1977; Brown and Day, 1983). A recent comparison of summaries from human experts versus extractive summarizers on forty-six topics from the TAC 2010 summarization challenge used automatic caseframe analysis, and found essentially these same properties in the human summaries, and not in the extractive ones (Cheung and Penn, 2013). Abstractive summarizers, however, are beginning to replicate the first two of these behaviors, as illustrated in many published examples based on encoder-decoder and pointer-generator neural architectures (Nallapati et al., 2016; See et al., 2017; Hsu et al., 2018; Guo et al., 2018). Summarization evaluation relies almost exclusively on ROUGE (Lin, 2004), an automated tool that cannot directly assess importance of summary content, or novel wording for the same information. We present an automated method to assess the importance of summary content, independent of wording, based on a widely used manual evaluation called pyramid (Nenkova et al., 2007).

Aligned PyrEval (W=5) and Manual (W=4) SCU	
RSUM1	**For example, an art gallery in London held an exhibit. with digital curr. as the preferred . . .**
RSUM2	However, there has been some positive **news as bus. such as a Scottish Hotel & a London Art Gallery are allowing cust. to pay with crypto currencies**
RSUM3	Cellan-Jones (2018) writes recent days **both a London art gallery and a Scottish hotel . . . to allow their cust. to pay with crypto-currencies.**
RSUM4	by suggesting that {a London art gallery & Scottish hotel chain plan to . . . support for different crypto-currencies.}$_{Paraph}$ . . . {*that the London based art gallery would use only crypto currencies*}$_{Paraph}$
RSUM5	*Businesses located in London and Scotland have made enquiries to allow payment from customers using cryptoc.*

Match to a student summary that used synomyms:
a *craftsmanship* exhibition alongside a Scottish *inn* have plans for their clients to pay in digital currencies

Figure 1: Alignment of a single PyrEval SCU of weight 5 to a manual SCU of weight 4 from a dataset of student summaries. The manual and automated SCUs express the same content, and their weights differ only by one. For each of five reference summaries (RSUM1-RSUM5), exact matches of words between the PyrEval and manual contributor are in bold, text in plain font (RSUM2, RSUM4) appears only in the manual version, and text in italics appears only in the PyrEval version. Paraphrases of the same content from RSUM4 were identified by human annotators (plain font) and PyrEval (italics). Also shown is a matching segment from a student summary, where the student used synonyms of some of the words in the reference summaries.

The pyramid method and ROUGE both use multiple summaries written by humans as references to assess new summaries. The manual pyramid method requires human annotators to identify Summary Content Units (SCUs) by grouping phrases from different reference summaries into

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 404–418
Hong Kong, China, November 3-4, 2019.

the same SCU if they express the same propositional content. Figure 1 illustrates an SCU from a manual pyramid applied to college student summaries of articles on cryptocurrency, with contributions from four of the five reference summaries (RSUM1-RSUM4). It is aligned to a nearly identical SCU constructed by PyrEval, with a contribution from the fifth reference (RSUM5). Previous work has shown that these kinds of discrepancies occur between human annotators, and have little effect on interannotator agreement or rankings of summarizers (Passonneau, 2010). The importance of an SCU increases with the number of reference summaries that express it, as indicated by its weight. If an evaluation summary expresses the same content as an SCU, its score is increased by the SCU weight (details below). ROUGE allows the user to select among numerous ways to measure ngram overlap of a new summary to the references, e.g., for different ngram sizes with or without skips, and with or without stemming. ROUGE does not, however, consider the relative importance of content, or account for synonyms of words that appear in the reference summaries.

We present PyrEval,[1] which outperforms previous work on automated pyramid in accuracy and efficiency. It produces human-readable pyramids, and prints matches between SCUs and evaluation summaries, which can support feedback for educational applications. PyrEval performs well on a new dataset of student summaries, where we applied the pyramid annotation. We also present results for TAC 2010 automated summaries, one of the more recent years where NIST applied pyramid evaluation. While ROUGE-2 more accurately identifies system differences than PyrEval, its performance is more sensitive to different topics.

2 Pyramid Content Analysis

The challenge in evaluation of summary content is that different equally good human summaries have only partial content overlap. van Halteren and Teufel (2003) annotated factoids (similar to FOL propositions, and to SCUs) for fifty summaries of a Dutch news article, and found a Zipfian distribution of factoid frequency: a small number of factoids represent 80% of the content in summaries, but a very long tail of rare content accounts for 20%. Pyramid annotation of ten summaries for a few DUC 2003 topics had a similar a Zipfian distribution (Nenkova and Passonneau, 2004).

Pyramid has had extensive reliability testing. A sensitivity analysis showed four reference summaries were sufficient for score reliability, and with probability of misranking errors below 0.01% (Nenkova and Passonneau, 2004; Nenkova et al., 2007). Interannotator agreement using Krippendorff's alpha on ten pyramids ranged from 0.61 to 0.89, and averaged 0.78 on matching new summaries to pyramids for 16 systems on 3 topics each (Passonneau, 2010). Comparison of two manual pyramid evaluations from distinct annotators showed that different pyramids for the same topic yield the same system rankings, even though SCUs from different pyramids typically do not align exactly (Passonneau, 2010).

The default size of a phrase that contributes to an SCU is a simple clause, but if it is clear from the context that a summary essentially expresses the same content expressed in other reference summaries, it is said to contribute to the same SCU, and the annotator must select at least a few contributing words. SCU weights reflect how many of $\mathcal{N}$ reference summaries express the SCU content. As such, SCUs are constrained to have no more than one contributor phrase from each reference summary. If a summary repeats the same information, the repetition will increment the count of total SCUs within one summary, but cannot be a distinct contributor. For example, the paraphrases from RSUM4 shown in Figure 1 add two to the total SCU size of the summary, but can only be used once to increment an SCU weight. Simple clauses in an evaluation summary that do not match pyramid SCUs add to the summary's SCU count, but have zero weight.

3 Related Work

Summarization is an important component of strategy instruction in reading and writing skills (Graham and Perin, 2007), but is used less than it could be due to the demands of manual grading and feedback. However, integration of NLP with rubric-based assessment has received increasing attention. Gerard et al. (2016) and Gerard and Linn (2016) applied automated assessment using rubrics to successfully identify students who need the most help, and facilitate and meaningful classroom interactions. Agejev and Šnajder (2017) used ROUGE and BLEU to identify col-

[1] Available at https://github.com/serenayj/PyrEval

lege students' L2 skills. Santamaría Lancho et al. (2018) used G-Rubric, an LSA-based tool, to help instructors grade short text answers to open-ended questions. Passonneau et al. (2018) found a high correlation of an automated pyramid with a manual rubric on a small set of summaries; see last paragraph of this section.

ROUGE is the most prevalent method to assess automated summarization. In 39 long papers on summarization in ACL conferences from 2013 through 2018 (mostly abstractive), 87% used ROUGE-1, ROUGE-2 or other variants such as LCS (longest common subsequence). A few used POURPRE (question answering) (Lin and Demner-Fushman, 2006), or METEOR (MT) (Denkowski and Lavie, 2014) to investigate scores based on weighted content or synonomy. POURPRE relies on string matching against reference units called answer facts, weighting matches by inverse document frequency. METEOR aligns words between reference and candidate, and can use relaxed word matching, such as WordNet synonymy. Despite its dominant use in previous work, Graham (2015) noted that the large range of ROUGE variants causes inconvenience and instability in evaluating performance. Graham's results from testing the 192 variants on DUC2004 data suggest that the ROUGE variants that correlate best with human evaluations are not often used.

PyrEval differs from other automated pyramid tools in its focus on accurately isolating and weighting the distinct SCUs in the reference summaries. Three previous semi-automated pyramid tools used dynamic programming to score summaries, given a manual pyramid (Harnly et al., 2005; Passonneau et al., 2013, 2018). The first of these used unigram overlap to compare summaries to a pyramid. Absolute scores were much lower than ground truth, but average system rankings across multiple tasks were accurate. A subsequent extension that used cosine similarity of latent vector representations of ngrams and SCUs, based on (Guo and Diab, 2012), had much better performance (Passonneau et al., 2013). This was extended further through use of a weighted set cover algorithm for scoring (Passonneau et al., 2018). PEAK was the first fully automated approach to construct a pyramid and score summaries (Yang et al., 2016). It uses OpenIE to extract subject-predicate-object triples from references, then constructs a hypergraph with triples as hyperedges. Semantic similarity between nodes from distinct hyperedges is measured using ADW's random walks over WordNet (Pilehvar et al., 2013), to assign weights to triples. On a small set of summaries used here in Table 1, PEAK raw scores had a high correlation with a manual summary rubric. PEAK was also tested on a single DUC 2006 topic, where the input text was first manually altered. Because PEAK is slow, Peyrard and Eckle-Kohler (2017) reimplemented it's use of the Hungarian algorithm to optimize their summarizer. Because PEAK produces many noisy copies of the same SCU, its output cannot be used to justify scores based on the unique matches or misses of a student summary to SCUs. Its score normalizations are inaccurate, and the un-normalized scores are impractical for general-purpose evaluation.

4 PyrEval System

To construct a pyramid, humans identify contributor segments[2] and group them into SCUs. Evaluating a summary is a simpler process of matching phrases to existing SCUs. PyrEval performs analogous steps, as shown in Figure 2. It first decomposes sentences of reference summaries (RSUM) into segments (DECOMP PARSE) and converts them into semantic vectors (LATENT SEM). It then applies EDUA to group the segment vectors into SCUs. EDUA (see below) is a novel restricted set partition algorithm that maximizes the semantic similarity within SCUs, subject to SCU constraints. Evaluation summaries (ESUM) are preprocessed in a similar fashion to convert them to segments represented as vectors. As in (Passonneau et al., 2018), PyrEval applies WMIN (Sakai et al., 2003) to find the optimal set of matches with pyramid SCUs. The remainder of this section describes each step.

4.1 Sentence Decomposition

The decomposition parser takes as input a phrase structure parse and dependency parse for each sentence, using Stanford CoreNLP (Manning et al., 2014). Every tensed verb phrase (VP) from the phrase structure parse initializes a new segment. The head verbs of tensed VPs are aligned to the dependency parse, and their dependent subjects are then attached to the segments. Words other than

[2]These can be discontinuous substrings, and can reuse words from other contributors, e.g., subjects of VP conjuncts.

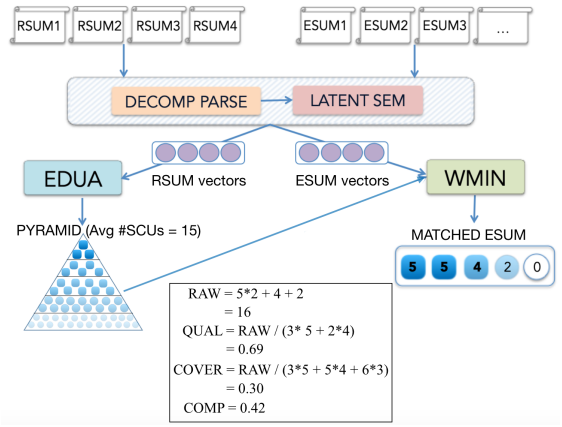

Figure 2: PyrEval preprocessors segment sentences from reference (RSUM) and evaluation (ESUM) summaries into clause-like units, then convert them to latent vectors. EDUA constructs a pyramid from RSUM vectors (lower left): the horizontal bands of the pyramid represent SCUs of decreasing weight (shaded squares). WMIN matches SCUs to ESUM segments to produce a raw score, and three normalized scores.

Data (size)	ELMo	USE	GloVe	WTMF
WIM (20)	0.3873	-0.0290	0.7149	**0.8525**
TAC 09 (54)	0.0515	0.2672	0.1713	**0.4961**
STS-14 (3750)	0.1636	0.5757	0.6129	**0.7257**

Table 1: Comparison of phrase embedding methods on correlation to manual pyramd (WIM, TAC09) or correlation to human similarity judgements (STS-14).

those in the VP and subject are reinserted in their original order. Every sentence has at least one default segmentation corresponding to the full sentence, possibly with one or more alternative segmentations of at least two segments each. It performs well for most cases apart from sentences with coordinate structures, which are notoriously difficult for conventional parsers. Figure 3 illustrates a sentence segmentation, with three alternatives.[3]

4.2 Semantic Vectors for Segments

The second PyrEval preprocessing step converts segments to semantic vectors. We chose to avoid semantic representation that requires training, to make PyrEval a lightweight, standalone tool. Although recent contextualized representations perform very well on a variety of NLP tasks, they are typically intended as the basis for a transfer learning approach, or to initialize further task-specific

[3]Segmentation 1.6.2 is the one EDUA-G selects for the pyramid.

Christopher Shake, the director of the London art gallery, suggests that this is not the case and that many different companies of different natures not just technology related are getting involved with cryptocurrencies.

1.6.1.0 that this is not the case
1.6.1.1 Christopher Shake, the director of the London art gallery, suggests and.
1.6.1.2 that many different companies of different natures not just technology related are getting involved with cryptocurrencies
1.6.2.0 Christopher Shake, the director of the London art gallery, suggests that this is not the case and.
1.6.2.1 that many different companies of different natures not just technology related are getting involved with cryptocurrencies
1.6.3.0 Christopher Shake, the director of the London art gallery, suggests and that many different companies of different natures not just technology related are getting involved with cryptocurrencies.
1.6.3.1 that this is not the case

Figure 3: Segmentation output for a sentence from a reference summary for the "CryptoCurrencies" topic of our student summaries.

neural training (e.g., (Pagliardini et al., 2018; Peters et al., 2018; Devlin et al., 2018; Vaswani et al., 2017)). The most practical way to rely on completely pre-trained representations is to use word embeddings along with a method to combine them into phrase embeddings. Here we report on a comparison of ELMo (Peters et al., 2018) and the Universal Sentence Encoder for English (USE) (Cer et al., 2018) with two conventional word embedding methods, GloVe (Pennington et al., 2014) and WTMF (Guo and Diab, 2012).[4]

ELMo is character-based rather than word-based, relies on a many-layered bidirectional LSTM, and incorporates word sequence (language model) information. It was trained on billions of tokens of Wikipedia and news text. To create meaning vectors for strings of words, we use pre-trained ELMo vectors, taking the weighted sum of 3 output layers as the word embeddings, then applying mean pooling.[5] USE is intended for transfer learning tasks, based on Transformer (Vaswani et al., 2017) or the (Iyyer et al., 2015) deep averaging network (DAN). We create meaning vectors for word strings with the USE-DAN pretrained encoder.[6] We use the GloVe download for 100D vec-

[4]We do not show results for Word2Vec (Mikolov et al., 2013), where performance was similar to GloVe.

[5]We use ELMo module from https://github.com/allenai/allennlp/.

[6]https://tfhub.dev/google/universal-sentence-encoder/2.

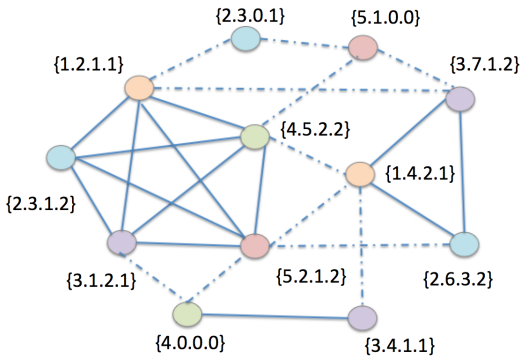

Figure 4: Part of an EDUA solution graph. Each vertex is a segment vector from a reference summary, indexed by **Summary.ID** (s_i)**, Sentence.ID** (s_{ij})**, Segmentation.ID** (s_{ijk})**, Segment.ID** (s_{ijkm}). All segments of all reference summaries have a corresponding node. All edges connect segments from different summaries with similarity $\geq t_{edge}$. This schematic representation of a partial solution contains three fully connected subgraphs with attraction edges (solid lines), each representing an SCU, whose weight is the number of vertices (segments).

1. Given a set of n reference summaries R, a preprocessing function (described in subsections 4.1-4.2) SEG returns segments as vectors: $\forall R_i \in R$, SEGS$(R_i) = \{seg_{ijk1}, seg_{ijk2}, \ldots, seg_{ijkm}\}$ where seg_{ijkm} is the mth segment of the kth segmentation of the jth sentence in the ith summary.
2. A graph G is constructed from SEGS(R_i), where an edge connects segments $seg_{ijkm}, seg_{i'j'k'm'}$ if $(i \neq i', seg_{ijkm} \neq seg_{i'j'k'm},$ cosine$(seg_{ijkm}, seg_{i'j'k'm}) \geq t_{edge})$. Every fully connected subgraph (clique) is a candidate scu whose size is the number of nodes, which has a maximum of n.
3. The attraction score of an $scu^z, \mathcal{AS}(scu^z) = \frac{1}{\binom{|scu^z|}{2}} \sum_{seg_{ijkm}, seg_{i'j'k'm} \in scu^z, seg_{ijkm} \neq seg_{i'j'k'm'}}$ cosine$(seg_{ijkm}, seg_{i'j'k'm'})$ if $z > 1$, else $= 1$.
4. A candidate pyramid P is a set of equivalence classes SCU^x that is a covering of all sentences in R (meaning only one segmentation per sentence belongs to any P), $\forall\ x \in [1,n] : (\exists\ SCU^x \in P) \rightarrow (x \in [1,n], \forall scu^z \in SCU^x, x = z)$. An SCU^x has an attraction class score $\mathcal{AC}(SCU^x) = \frac{1}{|SCU^x|} \sum_{scu^z \in SCU^x} \mathcal{AS}(SCU^z)$.
5. Finally, a pyramid P has an attraction score $\mathcal{AP}(P) = \sum_{SCU^x \in P} \mathcal{AS}(SCU^x)$.
6. The optimal pyramid(R) = $\mathcal{P}$ that maximizes $\mathcal{AP}$.

Figure 5: Formal specification of EDUA's input graph G consisting of all segments from all segmentations of reference summary sentences (item 2), the objective (item 6), and three scores for defining the objective function that are assigned to candidate SCUs (item 3), sets of SCUs of the same weight (item 4), and a candidate pyramid (item 5).

tors trained on the 840B Common Crawl.[7] To combine the GloVe word vectors into a phrase vector, we use the weighted averaging method from (Arora et al., 2016). WTMF is a matrix factorization method. We use WTMF matrices trained on the Guo and Diab (2012) corpus (393K sentences, 81K vocabulary size) that consists of WordNet, Wiktionary, and the Brown corpus.

We compare the four embedding methods on three datasets. Because our goal is to select a method that performs well on pyramid annotation, the first two datasets are human and machine summaries with manual pyramid annotations, with correlation of the manual pyramid and PyrEval scores as the metric. WIM (for *What is Matter*) is a dataset of student summaries with pyramid annotation from (Passonneau et al., 2018) with 20 student summaries on one topic. Note that PyrEval achieved a correlation of 0.85 on this data, compared with 0.82 for PEAK (Passonneau et al., 2018). We also use a subset of data from the NIST TAC 2009 summarizer challenge. We use summaries from all 54 peer systems on 14 of the 44 topics. We also use the STS-14 benchmark dataset of semantic similarity judgements (3750 sentence pairs), as in (Guo and Diab, 2012).

Table 1 shows WTMF to perform best on the three tasks by a large margin. We speculate this results from two factors. The lower dimensionality of WTMF vectors compared to ELMo or USE-DAN leads to higher maximum cosine values, thus better contrast between similar and dissimilar pairs. WTMF differs from similar matrix reduction methods in assigning a small weight to non-context words, which improves robustness for short phrases (fewer context words) Guo and Diab (2012). The authors also claimed that a training corpus largely consisting of definitional statements leads to co-occurrence data that is less noisy than sentences found in the wild.

4.3 EDUA

EDUA (Emergent Discovery of Units of Attraction) is a restricted set partition algorithm. It constructs an optimal pyramid to achieve the highest attraction (semantic similarity of segments) in all SCUs. Figure 4 schematically represents the input graph to EDUA (see also item 1 in Fig-

[7] https://nlp.stanford.edu/projects/glove/.

ure 5), whose nodes consist of the segment vectors described in the preceding section, and whose edges connect segments from different summaries whose cosine similarity $\geq t_{edge}$.[8] A candidate SCU is a fully connected subgraph (clique; item 2 in Figure 5). Every candidate SCU has an attraction score $\mathcal{AS}$ equal to the average of the edge scores (item 3 in Figure 5). A candidate pyramid is a set of SCUs that constitute a covering of all the sentences in the input reference summaries, with all segments for a given sentence coming from only one of its segmentations. The SCU weights for a pyramid, which are in $[1, n]$ for n reference summaries, form a partition over its segments, and each equivalence class (all SCUs of the same weight) has a score $\mathcal{AC}$ that is the average of its SCU scores (item 4 in Figure 5). The score for a candidate pyramid $\mathcal{AP}$ is the sum of its $\mathcal{AC}$ scores (item 5 in Figure 5). We use the sum rather than the average for $\mathcal{AP}$ to favor the equivalence classes for higher weight SCUs. The optimal pyramid maximizes $\mathcal{AP}$ (item 6 in Figure 5).

EDUA has two implementations. EDUA-C implements a complete solution based on depth first search (DFS) of candidate SCUs that guarantees the global optimum (max $\mathcal{AP}$). EDUA-G is a greedy approximation.[9]

4.4 EDUA-C

EDUA-C constructs an adjacency list that for each clique (candidate SCU) in the input graph, identifies all the other SCUs that satisfy two constraints: 1) for any given sentence, all SCUs reference the same segmentation; 2) all segments in all SCUs are distinct. DFS search proceeds through the adjacency list, ordering the SCUs by weight, until a path is found through all SCUs that meets the constraints. The solution has the highest $\mathcal{AP}$, or in the case of ties, the path found first.

Figure 6 illustrates a toy EDUA-C DFS tree. Each node depicts a candidate SCU clique, labelled by the number of nodes in the clique (SCU weight). No child node has a higher weight than its parent nodes. A child node is added to a search path (solid nodes) if it violates no constraints. Each of the six paths in the figure would receive an $\mathcal{AP}$ score. After DFS finds all legal paths, the one with highest $\mathcal{AP}$ is selected as the solution.

[8] The value of t_{edge} is automatically set to the 83rd percentile of all pairwise cosine similarities in the input data.

[9] See Appendix A for EDUA-G.

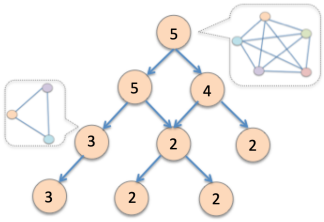

Figure 6: A directed Depth First Search tree for EDUA-C. Nodes are cliques representing candidate SCUs, as illustrated in Figure 4, labeled by their weights. Each DFS path is a partition over one way to segment all the input summaries and group all segments into SCUs. The solution is the path with the highest $\mathcal{AP}$.

4.5 Comparison of EDUA variants

Table 2 compares the distribution of SCUs by weight of the two EDUA variants with manual pyramids on the student summary dataset discussed in the next section. EDUA-C produces a more skewed distribution than EDUA-G. Both variants suffer from the coarse-grained segmentation output from the decomposition parser, but EDUA-G compensates by enforcing the Zipfian distribution observed in most pyramids (see appendix A for details).

To evaluate speed, we tested both variants on datasets with different numbers and lengths of reference summaries. TAC 2010 reference summaries (4 per topic) have on average 46 segments each, and 321 candidate SCUs. Pyramid construction for TAC 2010 takes less than 10 seconds with either variant on an Ubuntu machine with 4 Intel i5-6600 CPUs. EDUA-G's greater efficiency is more apparent for larger input. DUC 2005 has seven reference summaries per topic, and longer summaries than in TAC 2010; on five, EDUA-C takes 211 seconds, while EDUA-G is still only about ten seconds; on six, EDUA-C takes 20 minutes, compared to 5 minutes for EDUA-G.

Topic	Variant	All	w=5	w=4	w=3	w $\leq$ 2
CC	Manual	34	0	3	5	26
	G	31	1	2	4	24
	C	39	1	1	1	36
AV	Manual	41	0	6	2	33
	G	29	0	1	4	24
	C	35	1	1	1	32

Table 2: Comparison of distributions of SCUs by weight from pyramids produced manually, by two EDUA variants (G and C), for the two topics CC and AV.

4.6 WMIN Scoring

For automatic matching of phrases in evaluation summaries to SCUs in a manual pyramid, Passonneau et al. (2018) found good performance with WMIN (Sakai et al., 2003), a greedy maximum weighted independent set algorithm. Because EDUA pyramids are analogous to manual pyramids, PyrEval also uses WMIN. The input to WMIN is a graph where each node is a tuple of a segmentation of an ESUM sentence with the sets of SCUs that give the highest average cosine similarity for that sentence. The node weight is the sum of SCU weights. Graph edges enforce constraints that only one segmentation for a sentence can be selected, and each pyramid SCU can be matched to an ESUM sentence at most once. WMIN selects the nodes that result in the maximum sum of SCU weights for the ESUM.

Score computation is a function of the matched SCUs, as illustrated by the ESUM in the lower right of Figure 2. This ESUM has five SCUS: two of weight 5, one of weight 4, one of weight 2, and one that does not match the pyramid (zero weight). The sum of SCU weights is 16. The original pyramid score, a precision analog, normalizes the raw sum by the maximum sum for the same SCU count given by the pyramid – $(3 \times 5) + (2 \times 4)$ – indicating the degree to which the summary SCUs are as high weighted as possible. Following (Passonneau et al., 2018), we use the term quality score. The average number of SCUs in the reference summaries is 15, whose maximum weight from this pyramid is 53. Normalizing the raw sum by 53 gives a coverage score of 0.30 (a recall analog). The harmonic mean of these scores gives an F score analog referred to as a comprehensive score.

5 Student Summaries

As part of a collaboration with a researcher in educational technology, we collected a new data set of student summaries that were assigned in fall 2018 to computer science freshman in a university in the United Kingdom (Gao et al., 2019). Our immediate goal is to see how PyrEval could support instructors who assign summaries by providing an automated assessment that could be later corrected, but which provides scores and score justifications. PyrEval scores correlate well with manual pyramid scores on content, and the log output it produces provides a clear trace of score computation (see below).

Topic	Variant	Raw/Cov.	Qual.	Comp.	R2
AV	EDUAG	**0.66**	0.48	0.56	0.61
	EDUAC	0.55	0.50	0.53	
CC	EDUAG	**0.72**	0.63	0.69	0.66
	EDUAC	0.55	0.48	0.51	

Table 3: Pearson correlation of manual pyramid and PyrEval on four scores (raw/coverage, quality and comprehensive) compared with ROUGE-2 on coverage.

The class was an academic skills class that included instruction in academic reading and writing. For one assignment, they were instructed to select one of two current technology topics (three readings per topic), then to **summarize** it in 150 to 250 to words. The two topics are shown below, with the number of student summaries per reading, and average number of words.

1. Autonomous Vehicles (AV): 42 summaries, average words = 237.76
2. Cryptocurrency (CC): 37 summaries, average words = 245.84

To write reference summaries for both topics, the instructor recruited advanced students who had done well in her academic skills class in previous years. Three trained annotators applied manual pyramid annotation to the student summaries. As noted in section 2, pyramid annotation is highly reliable. Annotations of the student summaries were performed in two passes by different annotators.

Table 3 reports the correlation between the manual pyramid scores and the PyrEval scores on the two sets of student summaries. For both AV and CC, EDUA-G performs better than EDUA-C and ROUGE-2, the best ROUGE variant on TAC10 (see below), and ROUGE-2 performs better than EDUA-C. We attribute the lower correlations on the quality score, and the lower performance on this dataset compared to WIM (see Table 1), to the greater challenges of the new dataset. WIM students read a single, middle school text, and average summmary length was 109.02 words. For the new dataset, students read three advanced texts, and produced summaries that were over twice the length (see above). Error analysis shows complex sentence structure for the AV and CC data, with many constructions such as conjunctions and lists, that the decomposition parser cannot handle. As noted above, EDUA-G compensates due to a Zipfian constraint on the pyramid shape.

Figure 1 compares a PyrEval SCU with a manual one for the cryptocurrency topic, and

Single PyrEval SCU (W=3) about the relation of "car accidents" to "insurance cost"	
RSUM1	Also, as most collisions are due to human error, costs of insurance for self driving cars could fall by up to <NUM>.
RSUM2	The cars themselves would also reduce insurance premiums; <NUM> percent of road accidents are caused by human error
RSUM3	Shankleman does well to balance out the positives such as lower insurance , reduced traffic , savings on mechanical costs and lower chance of road accidents .

	Single manual SCU (W=4) on "high car accidents"	Single manual SCU (W=4) on "lower insurance"
RSUM1	Also, as most collisions are due to human error	costs of insurance for self-driving cars could fall by up to 50%
RSUM2	90 percent of road accidents are caused by human error	The cars themselves would also reduce insurance premiums
RSUM3	. . . lower chance of road accidents . . .	. . . lower insurance . . .
RSUM4	. . . he claims that over 90 percent of road traffic accidents occur as a result of human error	*this* would result in lower insurance premium for owners of autonomous vehicles by up to 50 percent.

Student IDs	Segments correctly matching this PyrEval SCU to students' summaries (from PyrEval log output)
A	The **insurance industry is also going to experience great changes** as the director insurer of AXA SA explains that more than <NUM> percent of **road accidents are caused by human error**.
B	as <NUM> of the **accidents are caused by human errors**, also reducing the number of human drivers will contribute to **cheap insurance premiums** and efficient transport
C	Shankleman explains how problems with modern day transport such as high **crash statistics and extortionate insurance costs will be eradicated** with such computing capabilities.

Figure 7: Alignment of an PyrEval SCU of weight 3 to segments from student summaries on autonomous vehicle.

also illustrates issues that might explain the relatively poorer performance of ROUGE. We show a phrase that both the manual annotator and PyrEval matched to the SCU from one of the student summaries, where the student used near synonyms for terms in the articles and reference summaries: *craftmanship* exhibition for *art gallery*, and *inn* for *hotel*. ROUGE cannot match synonyms, and does not distinguish differences in content importance.

Figure 7 shows an excerpt from PyrEval's log output on autonomous vehicle to illustrate the alignment of an SCU to three student summaries and comparison to two manual SCUs.[10] The PyrEval SCU captures a causal relation between "car accidents due to human error" and "lower insurance costs." The two manual SCUs, however, show that the human annotators split this content into two SCUs, because the content is expressed in distinct clauses in RSUM1 and RSUM2. The same content is supported by the implicit contexts for the shorter RSUM3 contributing phrases. The RSUM4 contributor in the manual SCU about "lower insurance" illustrates another issue that PyrEval preprocessing cannot handle: resolution of the deictic pronoun subject in "this would result . . . ".

6 TAC 2010 Summaries

NIST summarization challenges dealt exclusively with news, which is also the most prevalent genre for automated summarizers in our survey of 2013-2018 ACL publications (23/39 summarizers; see above). To evaluate ROUGE, NIST used two human gold standards in yearly challenges from 2005 through 2011, one of which was manual pyramid. Annotation was performed by volunteers among the challenge participants, using guidelines developed for DUC 2006.[11] In this section, we apply a method NIST helped develop to evaluate ROUGE against manual pyramid in an evaluation of PyrEval against manual pyramid. We selected TAC 2010 because summarizer performance was less good in the earlier years.

TAC 2010 had two 100-word summarization tasks on 10 documents for 46 topics. Task A summarization was guided by a query. Task B was an update to A, based on additional input. On inspection of the 92 pyramids (46 each for Tasks A and B), we found that roughly 27% had poor quality pyramids that did not follow the guidelines mentioned above. We assembled a team of five people familiar with manual pyramid to manually redo the twelve pyramids that were independently identified as the lowest quality.[12]

Tests of the correlation of human scores as-

[10] Preprocessing replaces numeric character strings with tags.

[11] http://www1.cs.columbia.edu/~becky/DUC2006/2006-pyramid-guidelines.html; we followed these guidelines for annotating the student

System	Task A			Task B		
	Mean Acc. (sdev)	95% CI	Acc. on 46 (delta)	Mean Acc. (sdev)	95% CI	Acc. on 46
R1	0.70 (0.04)	(0.69, 0.71)	*0.73 (0.03)*	0.61 (0.04)	(0.60, 0.62)	*0.69 (0.08)*
R2	**0.72 (0.03)**	(0.72, 0.73)	*0.80 (0.08)*	**0.70 (0.05)**	(0.69, 0.71)	*0.78 (0.08)*
EDUA-C	0.57 (0.04)	(0.57, 0.58)	0.65 (0.08)	0.56 (0.05)	(0.55, 0.57)	0.62 (0.06)
EDUA-G	0.57 (0.03)	(0.56, 0.57)	0.60 (0.04)	0.60 (0.03)	(0.59, 0.60)	0.63 (0.04)

Table 4: Mean accuracy, standard deviation and 95% confidence intervals on TAC 2010 Wilcoxon results for ROUGE-1, ROUGE-2 and PyrEval, using 100 bootstrapped samples of 41 of the 46 topics.

signed to automated summaries with ROUGE (and other automated metrics) were found to be unreliable, because of high score variance resulting as much from properties of the input texts as from differences in summarization systems (Nenkova, 2005; Nenkova and Louis, 2008). Analyses of over a decade of NIST data from automated summarizers that evaluate ROUGE against manual pyramid and another manual score led to a solution to this problem (Rankel et al., 2013; Owczarzak et al., 2012a,b; Rankel et al., 2011). The solution is to use Wilcoxon signed rank tests, so that pairs of systems are compared on matched input in a way that tests for statistical significance. The outcome is either that one of the systems is significantly better than the other, or that the difference between them is not statistically significant. To determine if the automated metric accurately reflects the gold standard scores, the same Wilcoxon tests are performed using the manually assigned scores on all pairs of systems, matching each pair on the same inputs. A given automated metric is then compared to the human gold standard to determine how accurately the automated metric leads to the same set of significant differences between all pairs of systems.

Table 4 presents bootstrapped accuracy results for ROUGE and PyrEval using 41 topics per bootstrap sample, along with absolute accuracy on all 46 topics. Each selection of 41 topics gives a gold standard set of system differences against which to compare a given metric. ROUGE 2 has the highest average accuracy on both Task A and B. ROUGE 1 performs nearly as well on Task A. PyrEval performs less well on average accuracy for all tasks, but similarly to ROUGE 1 in Task B. ROUGE-2 has greater sensitivity to topics, as shown by the higher deltas between the bootstrapped accuracy on 41 topics versus the accuracy on all 46. The differences in Table 4 between the bootstrapped averages across 41 topics, and the accuracy scores on all 46 topics, confirms the sensitivity of evaluation results to topics noted in (Nenkova, 2005; Nenkova and Louis, 2008).

7 Conclusion

PyrEval outperforms previous automated pyramid methods in accuracy, efficiency, score normalization, and interpretability. It correlates with manual pyramid better than ROUGE on a new dataset of student summaries, and produces output that helps justify the scores (similar to the examples for Figures 1 and 7). While it does not perform as well as ROUGE on extractive summarization, we speculate it would outperform ROUGE on abstractive summarizers. It relies on EDUA, a novel restricted set partition algorithm, that expects semantic vectors of sentence segments as input. The current rule-based method that identifies sentence substrings (the decomposition parser) is limited by the output of the constituency and dependency parsers it relies on. We are currently working on a neural architecture that simultaneously identifies simple clauses and produces semantic representations that could provide better input for both EDUA and WMIN, and thus improve PyrEval.

8 Acknowledgements

This work was supported in part by a Fellowship from Teaching and Learning with Technology, Penn State University, and by NSF award IIS-1847842. We thank two Penn State undergraduate research assistants for their contributions to the code base: Andrew Warner, and Purushartha Singh. Brent Hoffert, who recently graduated from Penn State, developed the wrapper that simplifies the use of PyrEval. Several additional Penn State undergrads helped correct the TAC 10 pyramids: Brent Hoffert, Alex Driban, Sahil Mishra, Xuannan Su, and Kun Wang. Finally, we thank the reviewers for their helpful suggestions.

summaries.

[12]We plan to ask NIST if we can make this data available through them.

References

Tamara Sladoljev Agejev and Jan Šnajder. 2017. Using analytic scoring rubrics in the automatic assessment of college-level summary writing tasks in l2. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 181–186.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR 2017)*.

Ann L. Brown and Jeanne D. Day. 1983. Macrorules for summarizing texts: The development of expertise. *Journal of Verbal Learning and Verbal Behavior*, 22:1–14.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1233–1242. Association for Computational Linguistics.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation (ACL)*, pages 376–380, Baltimore, MD.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Teun A. van Dijk and Walter Kintsch. 1977. Cognitive psychology and discourse: Recalling and summarizing stories. In W. U. Dressier, editor, *Trends in textlinguistics*, pages 61–80. De Gruyter, New York.

Yanjun Gao, Alex Driban, Brennan Xavier McManus, Elena Musi, Patricia Davies, Smaranda Muresan, and Rebecca J Passonneau. 2019. Rubric reliability and annotation of content and argument in source-based argument essays. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 507–518.

Libby Gerard, Marcia C Linn, and Jacquie Madhok. 2016. Examining the impacts of annotation and automated guidance on essay revision and science learning. In C. K. Looi, J. L. Polman, U. Cress, and P. Reimann, editors, *Transforming Learning, Empowering Learners: The International Conference of the Learning Sciences (ICLS) 2016*. Singapore: International Society of the Learning Sciences.

Libby F Gerard and Marcia C Linn. 2016. Using automated scores of student essays to support teacher guidance in classroom inquiry. *Journal of Science Teacher Education*, 27(1):111–129.

Steve Graham and Dolores Perin. 2007. A meta-analysis of writing instruction for adolescent students. *Journal of Educational Psychology*, 99(3):445–476.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–697. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th ACL*, pages 864–872.

Hans van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: Initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization*, pages 57–64. Association for Computational Linguistics.

Aaron Harnly, Ani Nenkova, Rebecca J. Passonneau, and Owen Rambow. 2005. Automation of summary evaluation by the pyramid method. In *Proceedings of the Conference of Recent Advances in Natural Language Processing (RANLP)*, pages 226–232.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, pages 74–81. Barcelona, Spain.

Jimmy Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9:565–587.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *20th SIGNLL Conference on Computational Natural Language Learning (CoNLL*, pages 280–290. ACL.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, pages 1436–1441. AAAI Press.

Ani Nenkova and Annie Louis. 2008. Can you summarize this? identifying correlates of input difficulty for multi-document summarization. In *Proceedings of ACL-08: HLT*, pages 825–833, Columbus, Ohio. Association for Computational Linguistics.

Ani Nenkova and Rebecca J. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *HLT-NAACL 2004*.

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

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012a. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karolina Owczarzak, Hoa Trang Dang, Peter A. Rankel, and John M. Conroy. 2012b. Assessing the effect of inconsistent assessors on summarization evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 359–362. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Rebecca J. Passonneau. 2010. Formal and functional assessment of the pyramid method for summary content evaluation. *Nat. Lang. Eng.*, 16(2):107–131.

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *ACL*, pages 143–147.

Rebecca J. Passonneau, Ananya Poddar, Gaurav Gite, Alisa Krivokapic, Qian Yang, and Dolores Perin. 2018. Wise crowd content assessment and educational rubrics. *International Journal of Artificial Intelligence in Education*, 28(1):29–55.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Maxime Peyrard and Judith Eckle-Kohler. 2017. Supervised learning of automatic pyramid for optimization-based multi-document summarization. In *ACL 2017*, pages 1084–1094.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *ACL*, pages 1341–1351.

Peter Rankel, John M. Conroy, Eric V. Slud, and Dianne P. O'Leary. 2011. Ranking human and machine summarization systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 467–473, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–136, Sofia, Bulgaria. Association for Computational Linguistics.

Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. 2003. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2):313–322.

Miguel Santamaría Lancho, Mauro Hernández, Ángeles Sánchez-Elvira Paniagua, José María Luzón Encabo, and Guillermo de Jorge-Botana. 2018. Using semantic technologies for formative assessment and scoring in large courses and MOOCs. *Journal of Interactive Media in Education*, 2018(1).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Qian Yang, Rebecca J. Passonneau, and Gerard de Melo. 2016. PEAK: Pyramid evaluation via automated knowledge extraction. In *AAAI*, pages 2673–2680.

A EDUA-G

EDUA-G (*Greedy*) is a greedy approximation to EDUA-C with a backtracking algorithm adjusting the allocation of candidate SCUs, and enforcing the constraints. In this appendix, we use the same notation as in section 4.3. Instead of finding the solution globally with maximum $\mathcal{AP}$ across all possible pyramids, EDUA-G works on achieving the maximum $\mathcal{AS}$ for each set of SCUs of a given size locally, starting with the set C_n with highest weight (number of nodes per subgraph c), then the rest in descending order. In addition to the constraints 1 and 2 in EDUA-C (mentioned in section 4.4), EDUA-G has a capacity constraint for each set C_r during search, limiting the number of SCUs committed to the class. This constraint is determined by the length of all the reference summaries and exploits an empirical observation of pyramids: that SCUs have a Zipfian distribution of frequency across reference summaries: a few have the highest weight, and for each lower weight there are more in number, with a very long tail of SCUs of weight 1.

To enforce the capacity constraint during search, we define the maximum number of SCUs y_n of each equivalence class C_n as:

$$y_n = \alpha\left(\frac{1}{n}\right)^{\beta} \tag{1}$$

where n is the index of the equivalence class, α is a constant related to the total number of segments from all reference summaries, and β is a scaling parameter (Clauset et al., 2009). Thus in addition to t_{edge}, EDUA-G has the hyperparameters α and β. The capacities of the equivalence classes are monotone increasing as n decreases:

$$|C_n| \leq |C_{n-1}| \tag{2}$$

Summing over $|C_n|$ gives the size of the pyramid:

$$\sum_{i}^{N} |C_n| \leq \sum_{i}^{N} y_i \tag{3}$$

Algorithm 1 presents EDUA-G.

Initialization Similar to EDUA-C, a segment pool $SP = SEGS(R_1) \cup \ldots \cup SEGS(R_n)$ is first constructed from all the reference summaries to store segments and two status flags. The pool is accessible globally. For every segment seg_{ijkm}, two status flags are set:

Algorithm 1: EDUA-G

Data: Number of reference summaries n; a list CU of candidate SCUs ordered by weight r where $1 \leq r \leq n$, then by attraction score $\mathcal{AS}(CU^r)$; capacity of each equivalence class $y_1 \ldots y_n$ by formula 1; a segment pool SP, residuals L_1
Result: Pyramid P with equivalence classes $C_1 \ldots C_n$

1 Initialize $r = n$,
2 $C_r = \emptyset$, $P = \emptyset$, D_r as empty stack,
3 **while** $(r > 1) \wedge (|C_r| \leq y_r)$ **do**
4 push all candidate CU^r selected from CU into D_r sorted by attraction score in ascending order ;
5 **while** D_r *is not empty* **do**
6 pop e from D_r with maximum $\mathcal{AS}$;
7 **if** *notConflict*(C_r, e), *and* $\forall\ seg_{ijkm} \in e$ $seg_{ijkm}.commit == True$ *or* $seg_{ijkm}.commit == NotValid$, *and* $seg_{ijkm}.used == False$ **then**
8 Commit(C_r, e, SP) ;
9 **end**
10 **end**
11 **if** *P fails to meet any of the constraints* **then**
12 BackTrack$(C_r, y_r, C_{r+1}, P, SP)$
13 **else**
14 $P \leftarrow P \cup C_r$;
15 **end**
16 $r \leftarrow r - 1$;
17 Initialize new stack D_r and repeat line 3
18 **end**
19 **foreach** $segs_{ijkm} \in L_1$ **do**
20 **if** $segs_{ijkm}.commit == True$ *or* $segs_{ijkm}.commit == NotValid$ **then**
21 $C_1 \leftarrow C_1 \cup segs_{ijkm}$;
22 $segs_{ijk*}.commit = True$ in SP;
23 **end**
24 **end**
25 $P \leftarrow P \cup C_1$

1. segmentation status, denoted as $seg_{ijkm}.commit$: for all $seg \in SP$, $seg.commit$ will be initialized as $NotValid$; during EDUA-G, if a sentence is first used by a segmentation seg_{ijk}, all segments $seg_{ijk*}.commit$ are set to True, and all other $seg_{ijk'*}$ from this sentence seg_{ij} are set to as False

2. segment status, denoted as $seg_{ijkm}.used$: when initialized, $seg_{ijkm}.used$ is set to False; if segment seg_{ijkm} is used in an SCU, the status $seg_{ijkm}.used$ is set to True

A graph G is constructed from all segments. A list of candidate SCUs (fully connected subgraph) with weights r from n to 2 is exhaustively extracted from G. All the leftover segments with weight as 1 are stored as residuals denote as L_1, at default sorted by the index.

Allocation The allocation process proceeds top-down, iterating over descending values of r from

Algorithm 2: BackTrack

Input: Current set C_r that fails constraints, its size constraint y_r, equivalence class C_{r+1}, current result P, segment pool SP
Result: Adjusted pyramid P with equivalence classes $C_1 \dots C_n$

Initialize $l_{r+1} \leftarrow$C$_{r+1}$, $l_r = \emptyset$;
while $|C_r| \leq y_r$ **do**
  Sort $l_r + 1$ by $\mathcal{AS}$ in ascending order;
  pop e' from $l_r + 1$;
  decompose e' into CU_r where $CU_r = \{CU_{r1}, \dots, CU_{r\binom{r+1}{r}}\}$;
  $l_r \leftarrow l_r \cup CU_r$;
  foreach $e \in l_r$ **do**
    if *notConflict*(C_r, e), $\forall\ seg_{ijkm} \in e$ $seg_{ijkm}.commit == True$ *or* $seg_{ijkm}.commit == NotValid$, *and* $seg_{ijkm}.used == False$ **then**
      Commit(C_r, e, SP) ;
    end
  end
  Update P with new C_r;
  Update C_{r+1} by removing all CU_{r+1}s that have overlapping segments with SCUs in C_r ;
  Update P with new C_{r+1}, reset segment and segmentation status in SP;
  if *P fails to meet any of the constraints* **then**
    BackTrack(C_{r+1}, y_{r+1},C_{r+1+1},P, SP)
  end
  Break ;
end

Algorithm 3: notConflict

Input: current set C_r, a candidate SCU e,
foreach *segment* $seg_{ijkm} \in e$ **do**
  if $\exists\ c \in C_r$ *where* $seg_{ijkm} \in c$ **then**
    return False
  end
end
return True

n to 2. All candidate SCUs are ordered first by weight, then by descending $\mathcal{AS}$. Each set C_r is filled with all candidate SCUs of size r, where maximum $\mathcal{AS}(SCU^r)$ is selected greedily, until the capacity constraint is satisfied. Every SCU committed to C_r requires the segment status to be checked and updated. Then the residual segments are allocated to C_1 as in EDUA-C if the status of the segments permits. For $1 < r < n$, if the provisional pyramid violates any constraints, backtracking considers a provisional revision of C_{r+1} based on reallocating all the segments in each subset of C_{r+1} of size q, for q from 1 to the size of C_{r+1}, considering reallocations in order of descending values of $\mathcal{AP}$. The algorithm terminates when all the constraints are satisfied, no segments remain whose segmentation status is True and whose segment status is False.

Algorithm 4: Commit

Input: current set C_r, a candidate SCU e, segment pool SP
$C_r \leftarrow C_r \cup e$;
foreach $seg_{ijkm} \in e$ **do**
  Set $seg_{ijkm}.used = True$ in SP ;
  Set $seg_{ijkm}.commit = True$ in SP ;
end

Backtracking The backtracking algorithm proceeds bottom-up, from the current set C_r to C_n. Recall from section 4.3, every pair of segments in an SCU has an edge $\geq t_{edge}$; therefore an SCU with $r+1$ contributors can be decomposed into $\binom{r+1}{r}$ SCUs with r contributors. We utilize this property to ensure every set C_r satisfies the constraints. During the emergent search and allocation of SCUs, if a set C_r does not meet the capacity constraint, the backtracking process will be initiated for re-allocation by re-using the segments committed to SCUs in C_{r+1}, to compose new SCUs in C_r. As shown in Algorithm 2, while the allocation process selects SCUs with maximum attraction scores greedily, the backtracking takes a conservative approach of re-doing the commit decision by decomposing one SCU at a time in C_{r+1} with the least $\mathcal{AS}(SCU^r)$, and composing new SCUs with weight r for C_r. It proceeds recursively from r to n until the resulting P satisfies the constraints. This is because every SCU in C_{r+1} has higher importance than in C_r, and this minimizes the impact of the re-allocation step on $\mathcal{AP}$. The backtracking algorithm terminates after all the constraints are satisfied.

B Grid Search on Hyperparameters

Grid search was used to tune the EDUA-G hyperparameters. On DUC 2005 data, we used α in the range [|seg|+10,|seg|+50] where |seg| is the number of input segments, and $\beta \in [1, 3]$. To set t_{edge}, we compute pairwise similarities of all segment pairs from different summaries, and take t_{edge} as the value at percentile N, for $N \in [60, 87]$. The performance metric was correlation with manual pyramid on individual summarization tasks.

Table 5 of ANOVA on the hyperparameters shows that β and t_{edge} have strong impact, while α does not (we select $\alpha = 10$). A contour plot of all combinations of β and t_{edge} (Figure 8) gives two regions of high correlation: $\beta \in [2.5, 3]$, and

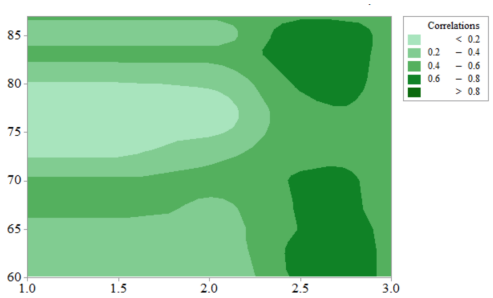

Figure 8: Contour plot for score correlations with β (X-axis) and t_{edge} (Y-axis).

Parameters	DF	F-value	P-value
α	4	0.31	0.872
β	4	104.31	0.000
t_{edge}	11	6.56	0.000

Table 5: One-way ANOVA for hyperparameters, with degrees of freedom (DF), F value and P-value (significance level α=0.05, sample size N=300).

$t_{edge} \in [60, 70]$, or $[80, 87]$. Higher t_{edge} yields fewer edges in the graph, so for efficiency, we select $\beta = 2.5$, and $N = 83$. (Depending on the dataset this corresponds to cosine similarities t_{edge} of about 0.15 to 0.35.)