The PYTHY Summarization System: Microsoft Research at DUC 2007

Kristina Toutanova, Chris Brockett, Michael Gamon,
Jagadeesh Jagarlamudi†, Hisami Suzuki and **Lucy Vanderwende**

Natural Language Processing Group
Microsoft Research
Redmond, WA 98052

†Multilingual Systems
Microsoft Research Laboratory
Bangalore, India 560080

{kristout,chrisbkt,mgamon,jags,hisamis,lucyv}@microsoft.com

Abstract

PYTHY is a trainable extractive summarization engine that learns a log-linear sentence ranking model by maximizing three metrics of sentence goodness: two of the metrics are based on ROUGE scores against model summaries and one is based on Semantic Content Unit (SCU) weights associated with sentences selected by past peers that were obtained during the Pyramid evaluations. In addition to sentences from the document set, our system considers simplified sentences for inclusion in the generated summaries. The feature weights of the model are optimized on the DUC 2005 data, with the final feature set for the submitted system being determined by ROUGE-2 scores against the DUC 2006 model summaries. For the DUC update task, the model was augmented with a novelty detection classifier.

1 Introduction

Over the past several years, the Document Understanding Conferences (DUCs) have enabled the research community to access a vital and growing body of data relevant to multi-document summarization tasks. These data consist not only of the primary DUC document collections and the associated manual summaries, but also data created though collaborative efforts by DUC participants such as the Pyramid annotations (Passoneau et al., 2005) and the SCU-marked corpus produced at the University of Ottawa from the results of the Pyramid evaluations in 2005 and 2006 (Copeck et al., 2006).

Part of the motivation of the Natural Language Processing Group at Microsoft Research in participating in DUC 2007 has been to investigate ways in which this important body of data might be leveraged to model extractive summarization. To this end, we built PYTHY (peer system 29), a summarization engine that allows us to utilize both the manual summaries and the Pyramid annotations available from previous years in training an extractive summarization system.

Given a set of documents and a topic description, PYTHY generates a summary of a specified length that maximizes the score of the summary. The scoring function is learned by fitting weights for a set of feature functions of sentences in the document set and is trained to optimize a sentence pair-wise ranking criterion. The scoring function is further adapted to apply to *summaries* rather than sentences and to take into account redundancy among sentences.

In Section 2 below we give an overview of the PYTHY system. In Section 3 we evaluate the contribution of different features and modeling choices made. Section 4 provides a short account of our effort to adapt PYTHY to the DUC 2007 update summarization task. Finally we conclude with a summary and discussion of our results in DUC 2007, and look ahead to future work.

2 The PYTHY System

PYTHY comprises a preprocessing component that supplies heuristically simplified sentences; a model for scoring sentences and summaries; and a search component to find the best summary. These are described in turn in the subsections below.

2.1 Sentence simplification heuristics

PYTHY includes a heuristically driven sentence simplification component that extends that in our DUC2006 system (Vanderwende et al., 2006). The basic principle remains the same: provide many useful sentence *alternatives* for the extractive summarizer to choose from, rather than deterministically shorten sentences before or after sentence selection. In 2007, the PYTHY preprocessor generates multiple simplified candidates per sentence, and deploys an expanded set of simplification operations.

In simplification, syntactic units output by a broad-coverage English parser (Ringger et al., 2004) are eliminated from the parse tree when the node matches heuristically determined patterns. (See (Vanderwende et al., 2006) for details of these patterns.) Since producing every combination of candidates with these patterns is likely to result in an explosion of simplified sentences, we adopt a two-step approach. First, nodes in the parse tree are tagged when appropriate with one of four levels of "deletability", ranked as follows:

Level 1: Adverbial phrasal modifiers: Adverbial and prepositional phrase modifiers (manner and time expressions)

Level 2: Adnominal modifiers: Noun appositives and nonrestrictive relative clauses

Level 3: Adverbial clausal modifiers: Gerundive clauses and past participials (sentence initial or preceded by a comma)

Level 4: Intra-sentential attribution: e.g., *X said that...*; *..., X reported Tuesday*

In the second step, the tagged nodes are deleted, with the following constraints: (1) all nodes assigned the same level of "deletability" are removed simultaneously, and (2) cross-level deletion is cumulative, i.e., removal of lower ranked nodes requires simultaneous removal of any higher ranked nodes that are also present in the sentence.

The second condition ensures that the following sentence, for example, produces just two simplified candidates, after deleting the tagged nodes:

> *[According to statistics from the Census Bureau and the Bureau of Indian Affairs,]*$_{Level1}$ *there are 1.43 million Indians [living on or near reservations.]*$_{Level3}$
>
> - *there are 1.43 million Indians living on or near reservations* (Applying Level 1 pattern)
> - *there are 1.43 million Indians.* (Applying Level 1 and 3 patterns)

The maximum number of simplified candidates generated per sentence is thus limited to 4. Additional candidates are created by splitting coordinated sentences into their individual conjuncts in a separate algorithmic path, but conjuncts do not currently go through further simplification (via deletion) operations.

Application of this sentence simplification algorithm to DUC 07 data affected 61% of all sentences in the original document clusters, yielding 1.38 candidates per sentence on average. Its effect is extensive in that almost 60% of all extracted sentences in our summaries are the product of the sentence simplification component. Simplification impacts Rouge scores positively under PYTHY, as seen in Section 3.

2.2 Sentence-level features in PYTHY

Our sentence features apply to sentences in the documents of the given clusters as well as to sentences which were derived from the original ones through simplification. When we discuss sentence features below, we will note any aspects which are specific to the treatment of sentence simplifications.

In the discussion below, we denote a sentence by s and its sequence of words by $w_1, w_2, \ldots, w_n$. A subsequence of the words in the sentence are judged to be content words and are denoted by $c_1, c_2, \ldots, c_m$. (We employ an extensive stop word list, also used in previous submissions, to determine the content words.) We denote the relative frequency of x by $\hat{p}(x)$. The exact collection of elements that this value is computed over are specified below for each feature.

The learned score for sentences is a linear function of the feature values:

$$Score(s) = \sum_{k=1\ldots K} w_k f_k(s)$$

All features are real-valued. The features fall into several natural groups:

- SumFocus features, including scores similar to those employed in SumBasic (Nenkova et al., 2006) and SumFocus (used for our 2006 submission (Vanderwende et al., 2006)). These features are defined as follows: SumClusterFrequency(s)=$\sum_{i=1\ldots m} \hat{p}(c_i)$. The relative frequency is computed over content words in the cluster. The simplified sentences are *not* included in the cluster for the purpose of computing relative frequency estimates. The SumClusterFrequency feature is the only feature used in SumBasic. The topic frequency feature is as follows: SumTopicFrequency(s)= $\sum_{i=1\ldots m} \hat{p_T}(c_i)$. Here the relative frequency $\hat{p_T}(c_i)$ is computed over the topic description.

- Other content word unigram frequency features: SumClusterHeadlineFrequency, which is a sum of the frequency of the content words in the collection of headlines in the cluster, and SumDocumentFrequency, which is a sum of the frequency of content words in the document (as opposed to the cluster).

- Sentence length features: binary features that fire if the length of the sentence is less than a given length (we used two features, for lengths five and ten).

- Sentence position features: SumDocumentStartFrequency and SumClusterStartFrequency, which are sums of the frequencies of content words in the collection of the first 100 words from the document and all documents in the cluster; a real-valued position feature; and binary features that fire if the position is less than a given value (we used five and ten).

- Beyond unigrams (bigrams, skip bigrams and multi-word expressions): SumClusterBigramFrequency, SumClusterMWEFrequency, SumClusterSkipBigramFrequency, and SumTopicSkipBigramFrequency. Bigrams are defined as consecutive content words. Multi-word expressions are named entities and other collocations determined by a rule-based grammar component (Ringger et al., 2004) and a lexicon. Skip bigrams are content word bigrams that occur with at most two intervening content words.

- Features of all tokens (rather than content words only): SumTopicTokenFrequency and SumClusterTokenBigramFrequency; these features are sums of relative frequency estimates for all tokens, including stop words.

- Features for simplified sentences: a binary feature firing if a sentence is a simplification, and a real-valued feature that indicates the ratio of the lengths of the simplified and original sentences.

- A full sentence feature: a binary feature that fires if the sentence has a verb (detected using dictionary lookup).

- An idf feature: a feature that indicates the sum of the inverse document frequencies of the content words in the sentences in the cluster. More formally, this feature is defined as follows: $\sum_{i=1...m} -log(numSent(c_i))$, where $numSent(c_i)$ is the number of sentences in the cluster in which the content word c_i occurs.

Length, frequency, idf, and/or position features have been used previously by other trainable summarization systems e.g. (Kupiec et al., 1995; Fisher and Roark, 2006; Yih et al., 2007).

Because our system is trained to rank sentences but the task is to choose a summary of a given length, we have a mismatch between training and testing conditions. To alleviate the problem, we use a modified version of some features in training to prevent longer sentences from always being preferred to shorter ones. For example, if the SumClusterFrequency feature has a positive weight, we would generally expect that longer sentences will receive higher scores. Therefore in training we redefine all features that are represented as a sum of per-word scores into a normalized sum, as follows: $f'(s) = \frac{f(s)}{numWords(s)}$. During runtime search we do not normalize the feature values for sentence length because the search algorithm takes into account the summary length constraint and we only need to compare summaries of the same length. In future work we would like to explore more principled solutions which learn to rank summaries.

2.3 Pair-wise ranking training criteria

To learn weights for the features we need to specify a training criterion. As noted in the introduction, we used annotated data from past DUC competitions to derive several goodness metrics for sentences. Each goodness metric lets us define a training criterion.

Each goodness metric of sentences specifies a set of preferences for sentence pairs, where one sentence in judged "better" than the other. The corresponding learning criterion seeks to assign higher scores to the "better" sentences. More formally, suppose that a goodness metric asserts a set of preferences for sentences: $\{ij : s_i > s_j\}$. Then our training criterion (objective) derived from this metric is as follows:

$$L(D) = \sum_{s_i > s_j} log(\frac{e^{\sum_k w_k f_k(s_i)}}{e^{\sum_k w_k f_k(s_i)} + e^{\sum_k w_k f_k(s_j)}})$$

The summation is over all comparable sentence pairs. This objective can be seen as learning to maximize the probability of choosing the better sentence from each pair of comparable sentences. A similar training objective has been used for ranking in text classification (Dekel et al., 2004) and information retrieval (Burges et al., 2005). In addition to this log-likelihood we add a quadratic regularization penalty on the model parameters.

To learn a model that uses several goodness metrics simultaneously, we combine the log-likelihood terms corresponding to each metric as follows: If the log-likelihood of metric M_l is $L_{M_l}(D)$, we define a combined criterion as follows: $L(D) = \sum_l L_{M_l}(D)$. This can be seen as maximizing the probability that we choose the "better" sentences from each comparable pair according to all criteria, assuming that we make the choices independently. It could be useful to fit separate weighting factors for each criterion.

In summarization we have multiple evaluation measures for summaries: linguistic quality, content responsiveness, Pyramids, ROUGE (Lin, 2004) scores, etc. Combining several criteria in training as we do is a way to address the requirement to score well across multiple evaluations.

Below we describe the three different goodness metrics we used. Note that the set of preferences each metric specified need not be complete (it could be a partial order). Thus different metrics give rise to a different number of comparable sentence pairs. Therefore, if computational resources are an issue, reducing the number of comparable pairs may be of interest and if the training set is very small, increasing that number may be desirable.

2.3.1 ROUGE oracle metric

The idea of this metric is to first select the best possible summary according to ROUGE from the set of sentences in the cluster and their simplified versions. All sentences in this ”oracle” summary are judged to be “better” than all sentences not in the summary according to this metric.

An oracle summary for a cluster and a topic is defined as the summary that has the highest average of ROUGE-2 and ROUGE-SU4 scores with respect to the model summaries. Since finding such an oracle summary presents a hard search problem, we used a greedy search on sentences for which we defined the following per-sentence scores:

$$w(s) = \frac{1}{Rank(s, R2)} + \frac{1}{Rank(s, RSU4)}$$

Here $Rank(s, R2)$ and $Rank(s, RSU4)$ denote the ranks of the sentence according to its sentence-level ROUGE-2 and ROUGE-SU4 scores. During search, sentences that had cosine similarity higher than a specified threshold vis-a-vis the already chosen sentences were discarded, to avoid redundancy.

2.3.2 Pyramid-derived metric

From previous years (DUC05 and DUC06), we have SCU annotations for some of the sentences proposed by peers. The University of Ottawa (Copeck et al., 2006) has organized the data such that for some of the sentences in the original document collection, a list of corresponding content units is known. For each content unit, a weight is also known from the Pyramid annotations. Thus we can define the score of a sentence to be the sum of weights of all content units present in the sentence. We further normalize this score to define the goodness of a sentence according to this metric:

$$w(s) = \frac{1}{numWords(s)} \sum_{c \in SCU(s)} weight(c)$$

This metric asserts that for every sentence pair where both sentences have been annotated with zero or more content units, $s_i > s_j$ if and only if $w(s_i) > w(s_j)$. Sentences that have not been selected by any peers do not participate in any preference relations. Note that our simplified sentences do not have any content unit annotations associated with them. We therefore assume that sentence simplifications contain all of the content units of the original un-simplified sentences for the definition of this metric.

2.3.3 Model Frequency Metrics

These are two metrics based on unigram and skip bigram frequency of words in the model summaries.

The unigram metric defines the goodness of a sentence as follows: $w(s) = \sum_{i=1 \dots m} \hat{p}_{models}(c_i)$. In other words, this is similar to the SumClusterFrequency value but the relative frequencies are computed over model summaries. The sum is over content words only. Note that we do not normalize for sentence length, which could have been advantageous. The bigram measure of goodness is similar, but instead of unigram frequency it looks at the sum of frequencies of skip content word bigrams in the model summaries. This is analogous to our SumClusterSkipBigramFrequency feature but is computed with respect using frequency in the model summaries. These two metrics aim to imitate ROUGE-1 and ROUGE-SU4. Each metric asserts that a sentence s_i is “better” than sentence s_j if $w(s_i)$-$w(s_j)> C$. The constant C here is a manually selected threshold which we chose by rough inspection, so that we do not generate too many sentence pairs. The thresholds used were 0.08 for unigram frequency and 0.018 for skip bigram frequency. In the experiments section below, we will see that a large number of training instances are generated, even at these threshold settings.

2.4 Dynamic sentence scoring

The system described so far assigns scores to sentences. When we generate a summary, we also need to deal with the problem of repetition of information. The problem is especially important for multi-document summarization and in a system such as ours, which considers multiple simplifications of the same sentence as candidates.

Our approach to modeling redundancy is similar in spirit to the SumBasic approach (Nenkova et al., 2006). We define a dynamic sentence score, which is the score of a sentence as a continuation of a given partial summary: $Score(s|prevS)$, where $prevS$ denotes a set of sentences in a summary prefix. The score of a complete summary consisting of sentences $s_1, s_2, \dots, s_p$ is defined as:

$$Score(s_1 \dots s_p) = \sum_{i=1 \dots p} Score(s_i | s_1 \dots s_{i-1})$$

If the summary prefix is empty, the score of a sentence is as defined before in Section 2.2. If the summary prefix is non-empty, the values of some features are discounted to avoid redundancy and the weighting function uses the modified feature values:

$$Score(s_i|prevS) = \sum_{k=1 \dots K} w_k f_k(s_i|prevS)$$

In particular, the values of all features that decompose as a sum of frequency estimates for words (content words, tokens, bigrams, skip bigrams, multi-word expressions) are discounted as follows: if the given word n-gram occurs in the summary prefix $prevS$, its frequency estimate is multiplied by a discount factor α. For example, the SumClusterFrequency feature is defined as follows:

$$f(s|prevS) = \sum_{i=1 \ldots m} disc(c_i, prevS)\hat{p}(c_i)$$

The value of the discount factor is: $disc(c_i, prevS) = \alpha$ if $c_i \in prevS$, and $disc(c_i, prevS) = 1$ otherwise. Similarly, for the SumClusterIDFFrequency feature, we update by adding a term, rather than multiplying by a factor, because that feature is in the log domain and multiplying the frequency by a factor α is equivalent to adding $log(\alpha)$ in log-space: $-log(p\alpha) = -log(p) - log(\alpha)$.

Similar updating was performed in (Nenkova et al., 2006) and (Yih et al., 2007). The main difference is that in PYTHY we have many more types of word frequency features. At present we have chosen to discount all frequency features in the same way , using the same discount factor α.[1] However, our experiments suggest that it will be beneficial to fit separate discount factors for different feature types. As it turned out, our system as submitted in the DUC 2007 main task did not perform the same discounting strategy for the features over all tokens: SumTopicTokenFrequency and SumClusterTokenSkipBigramFrequency. For these two features the system performed more conservative discounting (discounting only content word repetitions and not discounting stop word repetitions). This was unintended, yet it had a substantial positive effect on performance. This leads us to conclude that different discounting strategies may be applicable to different feature types.

2.5 Search

In our submission of 2006, the search for a highest-scoring summary was greedy and employed several heuristics. In our present submission, having defined scores for summaries according to the model, we attempt to perform a more exact search for the summary with the best score.

We employ a search algorithm based on a dynamic programming solution for the knapsack problem (McDonald, 2007). As in (McDonald, 2007), our search is not exact, because the mechanism for modeling redundancy induces global dependencies. We employ beams for each partial summary length to improve the accuracy of the search. We experimented with several different sizes of the beams and found that a beam size of five was enough to reach optimal ROUGE performance. When we used larger beam sizes, the search algorithm found summaries with better model scores, but this did not result in improved performance.

Feature Set	No Simplified		Simplified	
	R2	SU4	R2	SU4
SumFocus	0.078	0.132	0.078	0.134
PYTHY	0.089	0.140	0.096	0.147

Table 1: Performance of Pythy and systems including feature subsets with and without using simplified sentences. Results are ROUGE-2 and ROUGE-SU4 recall on all words on DUC06.

3 Experiments

In this section we discuss the performance of the system and analyze the contribution of the new features which were not available in SumFocus. We also analyze the performance of systems using different training criteria.

The models discussed were trained on the DUC2005 data, using the combination of three criteria discussed in Section 2.3. We report results on the DUC2006 data. Table 1 shows results for the SumFocus feature set and the complete feature set (PYTHY). For each of these sets, we report results with and without using sentence simplifications. When sentence simplifications are used, they are added as candidate sentences both in training and testing. For both feature sets we fit the discount factor α through optimizing ROUGE-2 on a subset of the DUC2005 data, using a grid search on α.

We should note that the SumFocus model shown in the Table is not equivalent to the SumFocus model of MSR's submission in 2006, which used a grid search against ROUGE-2 to select the relative weights of general and topic frequencies of words and a greedy search algorithm for finding summaries. The SumFocus model discussed here uses the learning and search algorithms presented above in Section 2.

The table shows that PYTHY significantly outperforms a system using only the SumFocus features. We can also see that while sentence simplifications have a small contribution to ROUGE when the minimal SumFocus feature set is used, their contribution is much larger for the full feature set of PYTHY. This is perhaps due to the increased ability of the feature set to judge the quality of sentence simplifications.

By using sentence simplification, we create more space to capture important content. The average number of sentences in a summary extracted by PYTHY is 17.04 when not using sentence simplification, while with simplification the number of sentences increases to 20.1.

To provide an intuition for the most important features of PYTHY, the following are the top seven features and their rounded weights: SumTopicFrequency (170), SumTopicTokenFrequency(81), SumHeadlineFre-

[1] We fit the discount factor α to maximize the ROUGE-2 score of the system on a subset of the training data.

Criterion	Num Pairs	Train Acc	ROUGE -stop		ROUGE all	
			R2	SU4	R2	SU4
Oracle	941K	93.1	0.076	0.107	0.093	0.143
SCUs	430K	62.0	0.078	0.108	0.086	0.134
ModelFreq	6.3Mln	96.9	0.076	0.106	0.096	0.147
All	7.7Mln	94.2	0.076	0.107	0.096	0.147

Table 2: Performance of PYTHY using simplified sentences, when different criteria are used for training. Results are ROUGE-2 and ROUGE-SU4 recall on content and all words on DUC06.

quency(42), SumStartClusterFrequency (17), SumSkipBigramTopicFrequency(17), SumClusterFrequency(14).

Next we analyze the performance of the PYTHY feature set when the weights are trained according to each of the three criteria of Section 2.3 and their combination. The results are shown in Table 2.

The second column in the table shows the number of training sentence pairs that each criterion generates. The SCU criterion generates the smallest number of comparable pairs, because less than half of the clusters have Pyramid annotations and because not many sentences in these clusters have content unit annotations. The ModelFreq criterion generates many more training pairs, because almost any two sentences are comparable according to it. Using all training criteria results in nearly eight million training instances.

The table also shows the training set accuracy of each model on the pair-wise criterion in the third column. That is, the fraction of sentence pairs, for which the “better” sentence received a higher score according to the model. We can see that these accuracies are generally high, except for the Pyramid derived criterion; it seems that the model is not able to learn the ranking according to content units. This could be either because the SCU annotations may be noisy or because the feature set we are using is not able to represent the ranking. The table also contains ROUGE scores for models trained using each criterion and the three in combination. We can see that even though the SCU-trained model achieves significantly lower scores on ROUGE measures against all words, its score is slightly higher for measures that ignore stop words.

Since we ultimately care about optimizing ROUGE as well as Pyramid content scores, we used a combination of all criteria for the final model PYTHY. It remains to be tested whether an SCU-trained model would result in a system with a higher Pyramid score.

4 Update Summarization

DUC 2007 included an update summarization pilot exercise in which the goal was to produce summaries of a set of documents given an already known/read set of documents within a set of topics. For each topic, the task was to generate three 100-word summaries, from each of three temporally-ordered document sets (A, B, C):

- a summary of set A (10 documents);
- a summary of set B (8 documents), given that the reader has already seen A; and
- a summary of set C (7 documents), given that the reader has already seen A and B.

PYTHY took part in the update summarization task as system 42.

4.1 Novelty classifier

To participate in this task the system was augmented by incorporating a score returned by a linear SVM novelty classifier that had been trained on TREC novelty track data from 2002 and 2003, a total of 23,846 sentences (Soboroff and Harman, 2003; Voorhees, 2004). The sentences from these two data sets are grouped into a total of 100 topics, and for each topic the sentences were presented for feature extraction (and classification) in sequence: Given the previously seen sentences within the topic (BG = background), each sentence s is labeled as being novel or otherwise. The SVM features are a combination of standard distance metrics and term overlap measures. The probability distance measures and the cosine similarity listed below use unigram word distribution representations of sentences and background.

- Kullback-Leibler divergence (both for s vs. BG and BG vs. s)
- Jensen-Shannon divergence
- Cosine similarity
- Number of content words in the intersection of BG and S (absolute and normalized by length of S)
- number of named entities[2] in the intersection of BG and S (absolute and normalized by length of S).

In training, word overlap features and Jensen-Shannon divergence were assigned the greatest weights by the classifier, followed by cosine similarity.

[2] Extracted from Wikipedia (www.wikipedia.org) by scanning all links and treating page titles as a named entities. Embedded links to pages are then counted as possible synonymous matches to the named entity.

We integrated the novelty detection classifier with PYTHY as follows. We used a weighting factor ν for the novelty model and multiplied the sentence scores of PYTHY by the probability according to the novelty model that the sentence is novel, raised to the power of ν. The dynamic sentence score of sentence s_i when extending a partial summary $prevS$ becomes as follows:

$$Score(s_i|prevS) = Score_{Pythy}(s_i|prevS)Pr(n(s_i|BG))^{\nu}$$

Here BG denotes the background document set. $Pr(n(s_i|BG))$ is a calibrated probability that the sentence s_i is novel and ν is an exponent used to trade off the relative influence of PYTHY and the novelty classifier. We estimated ν by hand, after eye-balling the results produced on the sample update task set, choosing a value of 1 for the submitted system.

5 Results in DUC 2007

5.1 Main task

The main summarization task in DUC 2007 required participants to generate 250-word summaries of 45 clusters of 25 newswire documents each in response to short questions about their content. PYTHY (as System 29) placed 2nd out of 30 systems participating in the ROUGE-2 evaluation, with an official ROUGE-2 score of 0.12028 that is significantly better (at the 95% confidence interval) than peer summaries that ranked 6th or lower. On this metric, the upper bound of the confidence interval overlaps with the lower bound of the confidence intervals of 5 human summarizers.

On ROUGE-SU4, PYTHY ranked 3rd, with a score of 0.17074, within the confidence intervals of the lowest scoring two human summarizers, and significantly better than peer systems that placed 8th or lower. In the human evaluations, PYTHY ranked fifth-equal in terms of Content. However, it placed only 24th in terms of Linguistic Quality, a mismatch that is primarily attributable to a low Grammaticality score stemming from issues in sentence simplification

PYTHY was also one of 11 peers that participated in the Pyramid evaluation. Here the system placed first equal in overall Pyramid score, and second in terms of mean Semantic Content Units (SCUs), with an average 11.19 SCUs per summary, and a middle-ranked 2.14 repetitions.

5.2 Update summarization pilot task

In contrast with the main task, the novelty-classifier-augmented PYTHY (peer system 42) performed less robustly, ranking an average 15th out of 24 participating peers, according to the combined ROUGE-2, ROUGE-SU4 and Pyramid scores. The system was within the 95% confidence intervals of systems ranked 10–18 by ROUGE-2 and those ranked 8–18 under ROUGE-SU4.

System	R-2	R-SU4
PYTHY+Novelty(1)	0.07135	0.11164
PYTHY+Novelty(.5)	0.07879	0.11929
PYTHY+Novelty(.1)	0.08721	0.12958
PYTHY	0.08686	0.12876
SumFocus	0.07002	0.11033

Table 3: PYTHY Performance on the Update Summarization Pilot Task

To understand the performance of the novelty-augmented PYTHY system better, we performed experiments using the DUC2007 update model summaries. We tested the system containing only SumFocus features, the PYTHY system, and PYTHY augmented with the novelty classifier, using values 1, 0.5, and 0.1 for the parameter ν. The weights used are shown in brackets in the system names in Table 3. The submitted system used a weight of $\nu = 1$, a parameter for which we had no labelled data on which to train.

Table 3 shows ROUGE-2 and ROUGE-SU4 scores obtained by the different systems. The results we report here are very slightly different (higher) than the official results from DUC07, because we use a non-jackknifed version of the ROUGE script to score models. We can see from the table that the optimal value for the weight of the novelty model is smaller than 1. The unaugmented PYTHY system, which ignores the background documents, outperforms our submitted version and achieves a performance which is more consistent with out performance on the main competition track. PYTHY augmented with the novelty classifier using a setting of $\nu = 0.1$ outperforms PYTHY.

Excessive weight assigned to novelty appears to have had the effect of biasing the system to sentences that are technically novel, but not necessarily relevant. This effect shows up particularly strikingly in some topics, namely the B clusters of 0711 and 7021, where the system identified low-relevance information as novel, resulting in Pyramid scores of 0.

5.3 Discussion

Overweighting of the novelty classifier feature may not be the only reason for PYTHY's degradation on the update summarization task. PYTHY uses a large number of features whose value distributions are sensitive to properties of the clusters, and the system may be brittle when those properties change.

The document clusters in DUC 2005 and DUC 2006 are reasonably large, averaging 700 and 920 sentences. The clusters of the DUC07 main task were smaller, with 540 sentences per cluster on average. The clusters in the update task consisted of few documents, and the documents were shorter, yielding an average of 180 sentences per cluster. The potential impact of these differences in data size should not be underestimated. For example, if the position feature in the training set takes values between 1 and 100, because documents normally have around 100 sentences, and in testing the same feature only has values between 1 and 15, the learned weight may be inappropriate when applied to the new data.

In similar vein, we believe that relative frequency scores estimated from small document collections may not be sufficiently indicative of topicality, because perhaps even the most on-topic terms occur only infrequently.

6 Post-DUC 2007: Future Work

PYTHY demonstrates that learning of sentence-level scoring functions using a combination of goodness metrics offers a promising line of investigation in extractive summarization. In the future we expect to explore ways to overcome the robustness issues presented to the system by changes in the distribution of the data. In addition, the grammatical quality of simplified sentences offers scope for improvement, and we will be attempting to better reconcile the competing demands of novelty and relevance in our novelty detection. We also expect to begin to address higher level challenges, such as learning features over whole summaries, as opposed to individual sentences.

Acknowledgements

This paper has benefited from discussions with Wen-Tau Yih and Chin-Yew Lin. Krysta Svore kindly provided us with the named entity list generated from Wikipedia. Galen Andrew provided useful comments on a draft version. In addition, we would like to thank members of the DUC community who contributed to the Pyramid evaluation. Our thanks also goes to members of the Butler-Hill group, especially Ben Gelbart, for their assistance with Pyramid creation and annotation.

References

Chris J.C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML 2005)*.

Terry Copeck, Diana Inkpen, Anna Kazantseva, Alistair Kennedy, Darren Kipp, Vivi Nastase, and Stan Szpakowicz. 2006. Leveraging DUC. In *Document Understanding Workshop (DUC 2006)*.

Ofer Dekel, Christopher Manning, and Yoram Singer. 2004. Log-linear models for label ranking. In *Neural Information Processing Systems (NIPS 2004)*.

Seeger Fisher and Brian Roark. 2006. Query-focused summarization by supervised sentence ranking and skewed word distributions. In *Document Understanding Workshop (DUC 2006)*.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Special Interest Group in Information Retrieval (SIGIR 1995)*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Workshop on Automatic Summarization at ACL 2004*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval (ECIR 2006)*.

Ani L. Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multidocument summarizer. In *Special Interest Group in Information Retrieval (SIGIR 2006)*.

Rebecca J. Passoneau, Ani Nenkova, Kathleen McKeown, and Sergei Sigelman. 2005. Applying the pyramid method in DUC 2005. In *Document Understanding Workshop (DUC 2005)*.

Eric K. Ringger, Robert C. Moore, Eugene Charniak, Lucy Vanderwende, and Hisami Suzuki. 2004. Using the Penn Treebank to evaluate non-treebank parsers. In *Conference on Language Resources and Evaluation (LREC 2004)*.

Ian Soboroff and Donna Harman. 2003. Overview of the TREC 2003 novelty task. In NIST *Special Publication 500-255: The Twelfth Text* RE*trieval Conference (*TREC *2003)*.

Lucy Vanderwende, Hisami Suzuki, and Chris Brockett. 2006. Microsoft Research at DUC 2006: Task-focused summarization with sentence simplification and lexical expansion. In *DUC 2006*.

Ellen M. Voorhees. 2004. Overview of TREC 2004. In NIST *Special Publication 500-261: The Thirteenth Text REtrieval Conference (*TREC *2004)*.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *International Joint Conference on Artificial Intelligence (IJCAI 2007)*.