Smart Initialization Yields Better Convergence Properties in Deep Abstractive Summarization

Liam Kinney*
Department of Computer Science
Stanford University
Stanford, CA 94305
liamk@cs.stanford.edu

Casey Chu
Institute for Computational Math and Engineering
Stanford University
Stanford, CA 94305
caseychu@stanford.edu

Maneesh Apte
Department of Computer Science
Stanford University
Stanford, CA 94305
mapte05@cs.stanford.edu

Abstract

Abstractive text summarization has been proposed as an alternative to the inherently limited extractive methods, but extant work is plagued with high training times. In this work, we introduce a set of extensions, including novel initialization techniques, that allow contemporary models to achieve 10x faster training time and comparable results. Our work also provides substantial evidence against the accepted evaluation metric for abstractive summarization, and establishes a speed benchmark for further research.

1 Introduction

Summarization is the task of compressing an article or passage into a shorter passage while forfeiting as little meaning as possible. Extractive summarization is summarization using only words from the input text, whereas abstractive summarization has an unlimited output vocabulary.

State-of-the-art performance for this task has been achieved by Nallapati et al. (2016), and previously by Rush et al. (2015), from whose work we took our baseline. The motivation to expand on the earlier publication is threefold: (1) the performance increase from Rush et al. (2015) to Nallapati et al. (2016) came at a great cost in training time (almost 5x for some models), (2) the boost in performance is marginal and based on a dubious and outdated evaluation metric Lin (2004), and (3) the model by Nallapati et al. (2016) incorporates a sequence-to-sequence model, and we wanted to effect low-level optimizations. The model introduced in Rush et al. (2015) features a simpler, homegrown neural language model, which can be tuned specifically for the task of text summarization.

We have extended the baseline model with a number of extensions to improve speed and maintain comparable performance. Due to the questionable reliability of ROUGE and a trend of high training times in extant models – those in Rush et al. (2015) took 4 days and one in Nallapati et al. (2016) took 18 – our priority was to improve time per epoch and convergence time. Faster summarization systems are required if future work in the field hopes to implement on any useful scale, and we hope our work provides a useful contribution.

*Code for news-funnel available on github: https://github.com/mapte05/news-funnel

Input (x_1, ... , x_n)	**Output** (y_1, ... , y_m)
Britain 's UN envoy on Wednesday urged stronger international support, including greater EU funding, for the African Union [AU] peacekeeping mission in Sudan's troubled Darfur region to improve security on the ground. ($n = 31$)	UN urges stronger EU support for AU peacekeeping force. ($m = 9$)

Table 1: Example input sentences and one of our generated summaries. The score of generating y_{i+1} is based on the context y_c as well as the input (x_1, ... , x_m). Note that our abstractive model allows for generalizing (ex. "Britain's UN envoy" to "UN") paraphrasing ("mission in Sudan's troubled Darfur region to improve security" to "force"), and compressing (dropping the "on Wednesday" and the "including greater EU funding") in the summaries. See Jing (2002) for a full list of operations inherent in summarization.

2 Baseline Model

Our baseline model, as in Rush et al. (2015), is an extension of a neural language model (Bengio et al., 2003) that conditions on the input (to-be-summarized) text and on the words output so far, as in a typical neural language model. Concretely, their model learns the probability distribution $p(y_{i+1}|x_1, \ldots, x_m, y_{i-C}, \ldots, y_i)$, where $x_1, \ldots, x_m$ is the input sentence, y_i is the ith word of the summary, and C is the context size. In the following, we abbreviate $x_1, \ldots, x_m$ as x and $y_{i-C}, \ldots y_i$ as y_c.

The model consists of two components, a *language model* component and an *encoder* component. They are combined in a final softmax layer:

$$p(y_{i+1}|x, y_c) \propto \exp(\mathbf{V}\mathbf{h} + \mathbf{W}\mathbf{e} + \mathbf{b}).$$

The language model component is a feedforward neural network with one hidden layer that takes the embedded context as input. It can be summarized as:

$$\mathbf{h} = \tanh(\mathbf{U}\mathbf{y}_c + \mathbf{c})$$
$$\mathbf{y}_c = \mathbf{E}[y_c],$$

where $\mathbf{E}[y_c]$ denotes a lookup in an embedding table containing all context words concatenated into one large vector.

The encoder component is a simple attention mechanism that takes a weighted average of the input embedded vectors smoothed over a window size of Q, conditioned on the context y_c:

$$\mathbf{e} = \sum_{i=1}^{m} p_i \bar{\mathbf{x}}_i$$
$$p_i \propto \exp(\tilde{\mathbf{x}}_i^T \mathbf{P} \tilde{\mathbf{y}}_c)$$
$$\bar{\mathbf{x}}_i = \frac{1}{2Q+1} \sum_{j=i-Q}^{i+Q} \tilde{\mathbf{x}}_j$$
$$\tilde{\mathbf{x}}_i = \mathbf{F}[x_i]$$
$$\tilde{\mathbf{y}}_c = \mathbf{G}[y_c],$$

where again $\mathbf{F}[x_i]$ and $\mathbf{G}[y_c]$ are embedding table lookups.

Rush et al. (2015) used a hidden layer size $H = 400$, embedding size $D = 200$, context size $C = 5$ and a smoothing window $Q = 2$. The model was trained using stochastic gradient descent on the cross-entropy loss, and the embedding tables were normalized before every epoch as a regularization measure, although no explicit regularization was introduced. At test time, a beam search was used to generate a likely summary from the learned probability distribution p.

The extensions we introduce to this model are in three parts: initializations that shorten convergence time, optimizations that shorten time spent per epoch, and postprocessing to clean up output.

3 Extension: Smart Initialization

Our main focus in improving the model was the time to convergence. We hypothesized that the convergence time could be reduced by initializing the model's parameters intelligently.

3.1 Pretrained Embeddings

Rush et al. (2015) do not mention how they initialized their word embedding matrices $\mathbf{E}$, $\mathbf{F}$, or $\mathbf{G}$. To accelerate the training of the model, we initialize these matrices to the GloVe embedding matrix $\mathbf{E}_0$ (Pennington et al., 2014), with each vector normalized to unit length for regularization purposes.

We were able to give a novel interpretation to the matrix $\mathbf{W}$, which multiplies the encoded vector $\mathbf{e}$ to produce a log-probability distribution over the vocabulary. Recall that the output of the encoder $\mathbf{e}$ is an average of the input sentence's embeddings, weighted with attention. We hypothesize that this component of the model is "extractive." In other words, this attention directly picks out which word to output as the next word. If this is the case, then the optimal initialization for $\mathbf{W}$ is $\mathbf{E}_0^T$, since the component of $\mathbf{E}_0^T\mathbf{e}$ with the highest value corresponds to the word whose embedding vector makes the smallest angle with $\mathbf{e}$ – namely, the word that is most similar to $\mathbf{e}$. Initializing the matrices in this way allows the model to take advantage of the embedding information so that it could, for example, treat synonyms similarly, even before training has begun,

This interpretation suggests that it would improve performance to have the encoder behave in an 'extractive' manner: to give attention to words that are likely to appear next in the output. We can achieve this with a smart initialization for $\mathbf{P}$.

Recall that $\exp(\tilde{\mathbf{x}}_i^T\mathbf{P}\tilde{\mathbf{y}}_c)$ is the attention given to the ith word. If we initialize

$$\mathbf{P} = [\mathbf{I} \quad \cdots \quad \mathbf{I}],$$

where the identity matrix $\mathbf{I}$ is repeated C times, then

$$\exp(\tilde{\mathbf{x}}_i^T\mathbf{P}\tilde{\mathbf{y}}_c) = \prod_{j=i-C}^{i} \exp(\tilde{\mathbf{x}}_i^T\tilde{\mathbf{y}}_c).$$

The GloVe vectors are trained so that $\tilde{\mathbf{x}}_i^T\tilde{\mathbf{y}}_c$ approximates the log-probability that $\tilde{\mathbf{x}}_i$ appears in the context $\tilde{\mathbf{y}}_c$, so with this initialization, the attention is distributed to the words in the input sentence approximately in proportion to how frequently $\tilde{\mathbf{x}}_i$ is expected to appear in the context given by $\tilde{\mathbf{y}}_c$.[1] Thus the model initially exhibits exactly the desired behavior, without any training whatsoever.

Inspired by this interpretation, we conjecture that the feedforward portion of the model serves as an "abstractive" component that complements the extractive component. This portion only takes the context – what the model has already outputted – and decides what to output next. If we initialize the first D columns of $\mathbf{V}$ with the columns of $\mathbf{E}_0^T$, then we can interpret the first D components of the first layer output $\mathbf{h}$ as the word vector of what the model wants to output next. We suspect that the matrix $\mathbf{U}$ learns grammatical rules; for example, when it multiplies vectors corresponding to contexts containing the phrase "in the", it outputs a prototypical word vector corresponding to nouns, indicating that nouns often follow the phrase "in the."

A future extension might add parameters that scale the relative contribution of the each component of the final layer, using a final layer of, for example,

$$\exp(\alpha_1\mathbf{V}\mathbf{h} + \alpha_2\mathbf{W}\mathbf{e} + \mathbf{b}),$$

and perhaps pre-tuning these values on the training set before actually training. This would allow the model to learn how abstractive or extractive to be. It may also prove useful to train each component separately.

[1] Unfortunately, the normalization step (used for regularization) perturbs this interpretation. Perhaps a different method of regularization woud avoid disturbing this interpretation.

3.2 Frequency Bias

To aid convergence, we initialized the softmax bias term $\mathbf{b}$ to the empirical log-probability distribution of the training set. That is, we counted the number of occurrences in the training set of each word in the vocabulary, smoothed the distribution with Laplace smoothing (added 1 to each frequency count), and initialized $\mathbf{b}$ with the log of this count. The intuition behind this change is as follows: when the output of the previous layers is around 0, the softmax final layer will still output a sensible distribution of words – proportional to their actual frequency – so that the network need not relearn this information during training.

3.3 Decaying Attention

This initialization is based on the observation that news is written with the most salient information at the beginning of the sentence. We can encode this observation in the network as follows. Recall that in the original model, $\exp(\tilde{\mathbf{x}}_i^T \mathbf{P} \tilde{\mathbf{y}}_c)$ is the attention given to the ith word. We added a bias term $\mathbf{d}$ as

$$\exp(\tilde{\mathbf{x}}_i^T \mathbf{P} \tilde{\mathbf{y}}_c + d_i)$$

with $\mathbf{d}$ initialized to $d_i = -\alpha i$ with $\alpha \geq 0$ to produce an exponential decay in the attention. We chose $\alpha = 0.22$, knowing that the training process would find the ideal weights.

4 Extension: Speed Improvements

In addition to improving the convergence speed, we also sought to reduce the wall-clock time it took for our model to run each iteration. The improvements explained in section 4.2 allowed all of our parameters to fit in GPU memory, and we were able to increase the batch size from 64 to 512 in order to process more samples at once.

4.1 Sampled Softmax

A well-known bottleneck in training networks NLP applications is the time required to backpropagate the cross-entropy loss throughout the entire vocabulary. Various methods have been proposed to approximate the training, one of which is the sampled softmax algorithm (Jean et al., 2014). The intuition behind this algorithm is to use only a subset of the vocabulary to compute an approximation to the final softmax layer so that the backpropagation needs only update weights for that subset. When many updates are applied, the result mirrors the behavior of full-vocabulary softmax. Every batch, we sample 2048 vocabulary words with a log-uniform distribution.

4.2 Reduced Vocabulary Size + Truncated Input

The task of summarization inherently involves sacrificing some data. Along these lines, we restricted the vocabulary size to the 75,000 most common words and replaced the rest with an unknown token, leaving fewer parameters to update.

We also observed that the meat of each article appears in the first half of the input sentence. To see this, note the following input sequences, in which the salient information is bolded.

> "**The Sri Lankan government on Wednesday announced the closure of government schools** with immediate effect as a military campaign against Tamil separatists escalated in the north of the country."

> "**Israel prepared Sunday for prime minister Yitzhak Rabin's state funeral** which will be attended by a host of world leaders, including us president Bill Clinton and the Jordanian and Egyptian heads of state."

> "**A Swedish U.N. soldier in Bosnia was shot and killed by a stray bullet on Tuesday** in an incident authorities are calling an accident, military officials in Stockholm said Tuesday."

To boost speed without sacrificing much salient content, our model truncates the input to a fraction of the longest input sequence (in our case, $\frac{1}{4} * 93 \approx 23$)

5 Extension: Postprocessing

After observing output from the baseline model, we noticed that the model made small readability errors that could be handled with simple processing. A few summaries contained incorrect sequential repetition (ex. "The sky was blue was blue on Monday") and occasionally terminated improperly in prepositions, articles, or conjunctions (ex. "The sky was blue on"), so we performed a simple postprocessing step on the output to remove such cases. This slightly improved ROUGE scores.

6 Experiments

We trained our models on the same training set that Rush et al. (2015) used, a filtered and preprocessed subset of English Gigaword (Graff and Cieri, 2003). Gigaword consists of news articles and their headlines from various international news sources – The headline acts as as summary of the article. For performance reasons, only the first sentence of the article is used, and preprocessing has removed articles where the first sentence clearly doesn't contain sufficient information to reconstruct the headline. The average input length was 31.3 words, and the average summary length was 8.3.

The dataset was partitioned into a training set of 3,803,957 pairs, a validation set of 1,536 pairs, and a test set of 1,951 pairs. We used exactly the same partitioning as Rush et al. (2015).

6.1 `rush-reimp`

The first model we trained was a reproduction of the model from Rush et al. (2015). Though the original was written in Torch, we reimplemented in TensorFlow with a few small modifications:

1. We initialized **E**, **F**, and **G** with GloVe embeddings.
2. We used an Adam optimizer instead of the simple stochastic gradient descent optimizer.
3. We used a learning rate of 0.005 with a exponential decay of $\frac{1}{2}$ every 10 epochs, instead of a learning rate of 0.05 halved whenever the validation accuracy does not decrease for an epoch. (Note that we used a cross-entropy loss *averaged* over each word in the minibatch of 64, whereas the original loss was summed.)
4. We truncated inputs (section 4.2) as well as restricted the vocabulary size to the 75,000 most common words for performance reasons.
5. We did not re-implement the extractive extension denoted ABS+.

We denote this model `rush-reimp`.

6.2 `samp-soft-1`

To improve the training time, we took all the changes from `rush-reimp` and added sampled softmax (section 4.1). We also implemented two of our smart initializations: frequency bias (section 3.2) and decaying attention (section 3.3). This model is denoted `samp-soft-1`.

6.3 `samp-soft-2`

For our third model, we implemented all of our smart initializations – most notably using the pretrained embeddings for **V** and **W** – as well as the tiled identity matrices for **P** (section 3.1). We also accelerated the learning rate decay to $\frac{1}{2}$ every 1.5 epochs, as we noticed characteristic signs of a too-high learning rate in previous runs. We denote this model `samp-soft-2`.

We trained all our models on a single Tesla M60 GPU using the early stopping criterion. This translated to $3\frac{1}{2}$ hours per epoch for our baseline implementation, and $1\frac{1}{3}$ hours per epoch for our fastest model. The wall-clock training time for `rush-reimp` was about 3 days, whereas our fastest

Model	GPU	hours/epoch	epochs until convergence	wall clock time
Rush et al.	unknown	2.6	15	4 days
Nallapati et al.	Tesla K40	10-12	14-16	6-8 days
`rush-reimp`	Tesla M60	3.5	21	3 days
`samp-soft-1`	Tesla M60	1.33	22.2	30 hrs
`samp-soft-2`	Tesla M60	1.33	4.4	5.8 hrs

Table 2: A comparison of the performance characteristics of the different models.

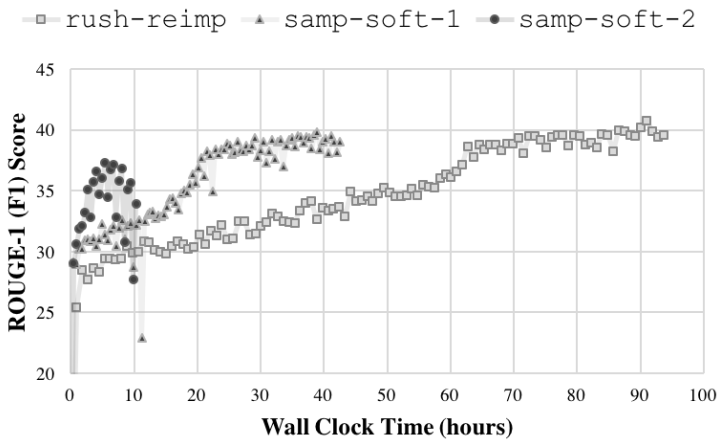

Figure 1: ROUGE score on validation set vs. wall clock time for our three models.

model `samp-soft-2` took about 6 hours. The full performance characteristics are listed in Table 2.

Because of the change of learning rate between `samp-soft-1` and `samp-soft-2`, it may be the case that the faster convergence of the latter model may be due to a change in learning rate rather than the smart initializations. A more in-depth investigation would attempt to isolate the cause of the faster convergence.

7 Evaluation

We evaluated our models on the held-out test set of 1951 Gigaword sentences, as well as on the DUC-2004 dataset (Over et al., 2007). DUC-2004 consists of 500 sentences (also from news stories) and 4 reference summaries. Note that their sentences and corresponding summaries are considerably longer than those of Gigaword.

We measured the performance of our model using ROUGE[2]. Whereas the more standard BLEU interpolates between different n-gram matches, there are several versions of ROUGE for different match lengths. We report ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest-common substring). There are also three variants of ROUGE, corresponding to recall, precision, and F1. The recall variant, while used more frequently in the past, suffers from a bias towards longer summaries, whereas the F1 variant does not have this deficiency (Nallapati et al., 2016).

When evaluating on the Gigaword test set, we report the F1 variant of ROUGE, running our model normally from start to finish. However, for the DUC-2004 dataset, we report the recall variant of ROUGE, evaluated on the first 75 bytes of output of our model, as is traditionally reported for this dataset. We also suppress the end-of-summary token so that our model produces as long a summary as possible (Nallapati et al. (2016)). Results are shown in table 3.

[2]Recall-Oriented Understudy for Gisting Evaluation

Model	Gigaword (F1)			DUC-2004 (R)		
	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
Rush et al.	29.78	11.89	26.97	28.18	8.49	23.81
Nallapati et al.	35.30	16.64	32.62	28.61	9.42	25.24
`rush-reimp`	28.95	12.38	26.70	23.97	6.80	20.66
`samp-soft-1`	28.35	11.70	26.25	23.54	6.59	20.60
`samp-soft-2`	27.82	12.03	25.95	23.35	6.67	20.43

Table 3: A comparison of ROUGE scores of the different models.

8 Analysis

8.1 Performance on Gigaword and DUC-2004

Our models falls just short of Rush et al. (2015) on the Gigaword test set and significantly worse on the DUC-2004 dataset. We suspect that this is due to our decision to truncate inputs to 23 words (section 4.2), which left our models incapable of incorporating information from the latter third of sentences. This truncation affected the sentences in DUC-2004 more drastically, since those sentences ran slightly longer on average (36 words compared to 31), had not been specially preprocessed to remove confusing sentences, and seem to be phrased in more roundabout ways. Additionally, it was difficult to feed untruncated sentences to our model only at test-time due to our addition of the attention decay bias term (section 3.3). To accommodate variable-length input sentences in the future, perhaps a model could train only the slope α of the attention decay. The trained value of the attention bias vector $\mathbf{b}$ was approximately linear with a slope of -0.03, suggesting that our choice of $-\alpha = 0.22$ was much too large.

8.2 Regularization and Learning Rate

An early training of our model indicated that the regularization procedure (normalizing the embedding matrices after each epoch) was critical to avoid overfitting to the training data. In fact, in the latter half of training of `rush-reimp` and `samp-soft-1`, the validation loss *increased* as training went on, only decreasing whenever the normalization step occurred. This suggests that the learning rate was too high at that point.

In `samp-soft-2`, the opposite problem occurred. The validation loss decreased as training went on, as expected, but went up during the normalization step; The increase was so drastic that the loss could not recover by the time the next normalization step occurred. This caused training to stop short of the local minimum. Perhaps reducing the frequency of the normalization step would help. Indeed, an explicit regularization term in the loss function would likely solve these issues as well.

8.3 ROUGE: A Poor Evaluation of Abstraction

After looking at the ROUGE scores of individual summaries, we argue that ROUGE is an inappropriate metric for evaluating abstractive summarization. Precision and recall, the bases of ROUGE, value similar word choice and ordering, but these are not the qualities that make a good summary. In fact, an effective summary contains different, more general terms than the original and does not necessarily mirror the original semantics.

Unfortunately, this is where `samp-soft-2` shines. Because it was initialized with the pretrained embeddings, it is more likely than the other models to output words in the *vicinity* of the extractive choice. It will regularly forego a word or phrase that appears in the original in favor of a synonym or generalization.

Consider example 1 in Table 4 and Table 5. To produce the output, `samp-soft-2` generalizes "Tehran" to "Iran", switches from passive voice to active voice, substitutes "frees" for "has been released from", and replaces "academic" with "scholar". These are valid operations and they produced a good, succinct summarization, but ROUGE penalized these differences and output a score of 0.

Example	F1 ROUGE-1 score		
	Rush et al.	samp-soft-1	samp-soft-2
1	52.6	57.1	**0.0**
2	42.1	58.8	13.3
3	62.5	**80.0**	71.4
4	10.0	26.1	22.2
5	37.5	11.8	30.8
6	25.0	28.6	28.6

Table 4: ROUGE scores of the examples in Table 5. Score indicates similarity to **Gold**

Consider example 3. The output from samp-soft-1 is syntactically incorrect – "death toll rises to school" makes no sense – but achieved the highest score in the table because it uses exactly the same words as the **Gold** summary. The output of samp-soft-2 is semantically proper but gets a worse score.

In examples 1, 4, 5, and 6, the worst summary got the top score, and in examples 1 and 3, arguably the best summary earned the lowest score. Not only is ROUGE often inaccurate; it regularly does the opposite of what's desired. Even though our models fall short of the state-of-the-art ROUGE scores, they might actually exceed in true performance.

A useful evaluation metric for abstractive summarization should prioritize symmetry of meaning rather than that of word choice or order. It should penalize wordiness and reward valid generalizations. ROUGE does none of these things, so it cannot be the standard for evaluating abstractive summarizations.

9 Conclusion

This was a particularly instructive project because of the simple link between the theory and the math behind the task. If we picture a journalist coming up with a headline for her article, we could draw the following analogies with our extensions:

1. Initializing with Pretrained Embeddings $\leftrightarrow$ Knowing the native language of the audience
2. Initializing with Frequency Bias $\leftrightarrow$ Knowing the standard language for headlines
3. Decaying Attention $\leftrightarrow$ Knowing that the most salient information comes at the beginning of the article

The abundance of knowledge available in the Information Age is a double-edged sword. The speed at which the human mind can ingest data is severely limited by today's computing standards, but the ability to condense information without sacrificing meaning is a solution. We hope that our work helps make this technology a reality.

Acknowledgments

We would like to thank Christopher Manning and Richard Socher, our instructors in CS 224N. We would also like to thank Kevin Clark, our final project advisor for the class. Final thanks to Microsoft for providing a GPU subscription through their Azure platform.

References

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

D. Graff and C. Cieri. English Gigaword. *LDC2003T05. Linguistic Data Consortium, University of Pennsylvania*, 2003.

Input(1):	a detained iranian-american academic accused of acting against national security has been released from a tehran prison after a hefty bail was posted , a top judiciary official said tuesday.
Gold:	iranian-american academic held in tehran released on bail
Rush et al.:	detained iranian-american academic released from prison after hefty bail
rush-reimp:	released from jail after posting bail
samp-soft-1:	detained academic released on bail
samp-soft-2:	iran frees detained scholar
I(2):	ministers from the european union and its mediterranean neighbors gathered here under heavy security on monday for an unprecedented conference on economic and political cooperation.
G:	european mediterranean ministers gather for landmark conference by julie bradford
R:	mediterranean neighbors gather under heavy security for unprecedented conference
rush-reimp:	eu mediterranean neighbors meet under heavy security
samp-soft-1:	eu mediterranean ministers gather for unprecedented conference
samp-soft-2:	eu ministers meet on security
I(3):	the death toll from a school collapse in a haitian shanty-town rose to ## after rescue workers uncovered a classroom with ## dead students and their teacher , officials said saturday.
G:	toll rises to ## in haiti school <unk>: official
R:	death toll in haiti school to ## dead students
rush-reimp:	death toll in haiti rises to # dead
samp-soft-1:	death toll rises to school in haiti
samp-soft-2:	death toll rises to # in haiti
I(4):	australian foreign minister stephen smith sunday congratulated new zealand 's new prime minister-elect john key as he praised ousted leader helen clark as a " gutsy " and respected politician.
G:	time caught up with nz 's gutsy clark says australian fm
R:	australian foreign minister congratulates smith new zealand as leader.
rush-reimp:	new zealand congratulates new zealand 's new zealand congratulates new zealand 's
samp-soft-1:	australian fm congratulates new zealand 's new zealand congratulates new zealand 's
samp-soft-2:	australian foreign minister congratulates new nz pm
I(5):	two drunken south african fans hurled racist abuse at the country's rugby sevens coach after the team were eliminated from the weekend's hong kong tournament , reports said tuesday.
G:	rugby union : racist taunts mar hong kong sevens : report
R:	south african fans racist abuse at rugby sevens tournament
rush-reimp:	south african fans racist abuse
samp-soft-1:	south african fans hurl firebombs at rugby world cup
samp-soft-2:	s. african rugby fans racist
I(6):	christian conservatives - kingmakers in the last two us presidential elections - may have less success in getting their pick elected in #### , political observers say .
G:	christian conservatives power diminished ahead of #### vote
R:	christian conservatives in the last two us presidential elections
rush-reimp:	conservatives may have less success
samp-soft-1:	conservative christian conservatives win in presidential elections
samp-soft-2:	christian right to vote in new hampshire

Table 5: Example sentence summaries produced on the Gigaword test set.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007, 2014. URL `http://arxiv.org/abs/1412.2007`.

H. Jing. Using hidden markov modeling to decompose human-written summaries. *Computational linguistics*, 28(4):527–543, 2002.

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang, et al. Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

P. Over, H. Dang, and D. Harman. DUC in context. *Information Processing & Management*, 43(6): 1506–1520, 2007.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.