Efficient Unsupervised Sentence Compression by Fine-tuning Transformers with Reinforcement Learning

Demian Gholipour Ghalandari[1,2]**, Chris Hokamp**[1]**,**
Georgiana Ifrim[2]
[1]Aylien Ltd., Dublin, Ireland
[2]Insight Centre for Data Analytics, University College Dublin, Ireland
[1]{first-name}@aylien.com
georgiana.ifrim@ucd.ie

Abstract

Sentence compression reduces the length of text by removing non-essential content while preserving important facts and grammaticality. Unsupervised objective driven methods for sentence compression can be used to create customized models without the need for ground-truth training data, while allowing flexibility in the objective function(s) that are used for learning and inference. Recent unsupervised sentence compression approaches use custom objectives to guide discrete search; however, guided search is expensive at inference time. In this work, we explore the use of reinforcement learning to train effective sentence compression models that are also fast when generating predictions. In particular, we cast the task as binary sequence labelling and fine-tune a pre-trained transformer using a simple policy gradient approach. Our approach outperforms other unsupervised models while also being more efficient at inference time.

1 Introduction

In general, the information content of text is correlated with its length. However, for a given text, a shorter version may still convey the essential information while preserving grammaticality (Siddharthan, 2014). The definition of *essential* can change depending on the downstream application, thus models for text compression must be able to adapt based on information about the downstream task.

Sentence compression models have been used as sub-modules of text and speech summarization (Banerjee et al., 2015; Shang et al., 2018), for headline generation (Dorr et al., 2003), subtitle generation (Vandeghinste and Pan, 2004), and summarizing emails (Zajic et al., 2008). Potential applications also include snippet generation and highlighting for social media, blog posts or search results.
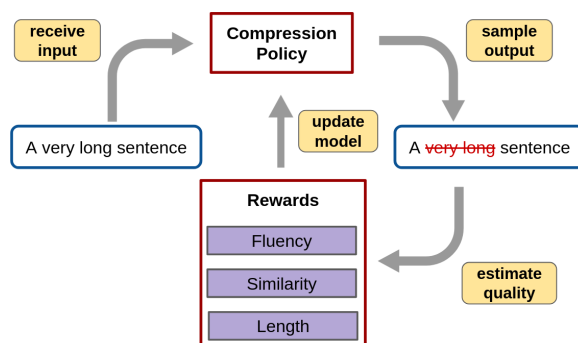

Figure 1: Reinforcement learning framework for sentence compression.

Given a particular text compression task, relevant evaluation metrics and auxiliary models of compression quality may not be straightforward to formulate as well-behaved differentiable objectives that can be used with standard backpropagation. In addition, ground-truth examples may be difficult to obtain because the annotation task is difficult to fully specify, and metrics which capture different facets of compression quality, such as fluency and optimal sentence length, may be negatively correlated. Even in the case where ground-truth examples are available, they are likely to represent only a subset of the possible outputs, so there is a risk of over-fitting or biasing models when relying solely upon a small amount of gold training data for optimization.

Recent unsupervised sentence compression approaches leverage powerful neural language models to directly optimize objectives such as fluency and faithfulness of compressed sentences, using discrete search strategies, without relying on ground-truth examples (Niu et al., 2019; Zhou and Rush, 2019; Schumann et al., 2020). However, these search-based methods are very inefficient at inference-time because the search must navigate through a large candidate space while recomputing expensive reward functions.
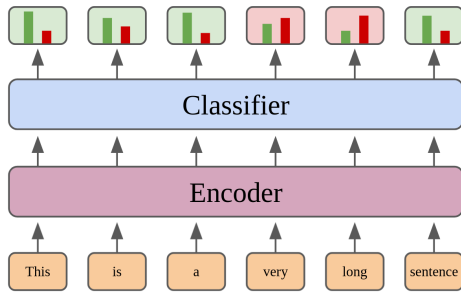
1267

Figure 2: Model architecture for compressing sentences.

To allow for flexible reward specification, while also enabling efficient inference, we design a simple and effective reinforcement learning (RL) setup: our model is initialized as an unsupervised pretrained language model with an untrained binary classification head (see Figure 2), and the sentence compression task is framed as sequence labeling, with optimization via policy gradient using a suite of reward functions. Sentences are compressed in an instantaneous, one-step fashion, similar to modern part-of-speech tagging or named entity recognition models. This approach simplifies the learning setup while also allowing for high throughput.

According to quantitative evaluation on several summarization benchmarks, our approach shows similar or superior performance compared to search-based methods, while also being much faster at inference time.

Our approach to unsupervised extractive sentence compression has the following benefits:

- **Unsupervised:** No labelled examples are required.

- **Fast inference:** At test time, the model only performs one-step sequence labeling.

- **Configurable:** Rewards can be tailored to specific use cases.

We review related work in Section 2. Section 3 formalizes the task. Section 4 gives a detailed description of the model and reward functions. Section 5 presents experimental results, and Sections 6 and 7 provide analysis and discussion of our findings.

2 Related Work

Unsupervised Sentence Compression

Early work on sentence compression casts the task as an optimization problem under linguistically motivated constraints (Hori and Furui, 2004; Clarke and Lapata, 2006, 2008). The objectives to be optimized include n-gram language model scores and frequency-based word relevance measures. Constraints are designed to ensure the grammaticality of compressions.

Some recent work follows the discrete optimization paradigm while leveraging powerful models as objective functions in place of hand-crafted constraints, while exploring different strategies for heuristic search: Zhou and Rush (2019) use beam search to optimize a fluency and a similarity objective. Schumann et al. (2020) use a greedy hill-climbing search to optimize fluency and similarity objectives. Niu et al. (2019) use a greedy search with a look-ahead mechanism, only optimizing fluency. All of these recent approaches use large neural language models to estimate fluency. While the approach presented in whis work does not involve discrete search, we consider it complementary and orthogonal to our RL-based approach (see Section 7 for more discussion).

Another commonly proposed unsupervised framework is to use autoencoders and reconstruction objectives (Miao and Blunsom, 2016; Févry and Phang, 2018; Malireddy et al., 2020). These approaches are based on the assumption that a good sentence compression is one from which the original sentence can be inferred.

Wang et al. (2018) is an example of prior work using reinforcement learning for unsupervised sentence compression. They use a Deep Q-Network to optimize a reward incorporating n-gram language model probabilities and grammatical constraints. This model repeatedly deletes a token until it terminates, as opposed to our one-step approach. Zhao et al. (2018) also use RL to optimize a syntax-focused language model score. However, their policy is initialized with a supervised sentence compression model, whereas ours is fully unsupervised.

Reinforcement Learning for Summarization

Reinforcement learning has become popular in the wider field of text summarization, finding applications in both extractive and abstractive sub-tasks. One use case of RL is in supervised scenarios, where rewards are computed based on ground-truth examples, e.g., ROUGE scores, to overcome issues with cross-entropy losses (Paulus et al., 2017; Narayan et al., 2018; Dong et al., 2018). BANDIT-SUM (Dong et al., 2018) in particular has a very similar RL setup to ours: they train in one-step episodes where a policy predicts extractive labels

and immediately receives a reward. Scialom et al. (2019) augment a ROUGE-based reward with a reward based on question answering. Böhm et al. (2019) and Stiennon et al. (2020) learn reward functions from human quality ratings of summaries. Similar to our unsupervised approach, Laban et al. (2020) use RL for unsupervised abstractive summarization, optimizing reward functions representing fluency, coverage under a length constraint, and also use a policy gradient approach.

3 Task

We focus on the specific task of summarizing a sentence by extracting a subset of its tokens in their original order. Given an input sentence x consisting of n tokens $x = (x_0, x_1, ..., x_n)$, we aim to produce a sequence of binary labels $y = (y_0, y_1, ..., y_n) \in \{0, 1\}^n$, where each label indicates whether the corresponding input token should be included in the compressed version of a sentence.

We further assume an objective function, or *reward* function $R(x, y)$ that measures how well applying the labels y summarizes the original sentence x. For a particular x, the goal is to find $\text{argmax}_y R(x, y)$, without access to any ground-truth examples.

In general, there are 2^n possibilities to shorten a sentence in this task. A fixed summary length L would reduce this to $\binom{n}{L}$ possibilities, peaking at $L = \frac{n}{2}$ (for even n). We do not constrain our approach to a fixed length, but we compare it to search-based techniques that are constrained to the $\binom{n}{L}$ search space.

4 Method

4.1 Training Procedure

We train a policy π_θ with parameters θ to produce binary labels. Given an input x, the policy π_θ predicts a binary keep/discard probability distribution for each token index in x. We use the notation $\pi_\theta(*|x)$ to refer to the collection of these distributions for all tokens in x. We obtain the probability $\pi_\theta(y|x)$ of a label sequence y given input sequence x as follows:

$$\pi_\theta(y|x) = \prod_i \pi_\theta(y_i|x), \quad (1)$$

where $\pi_\theta(y_i|x)$ is the probability of a token x_i being included if $y_i = 1$ or excluded if $y_i = 0$. To compress a sentence using π_θ, we select the higher scoring label for each token:

$$y^a = \{\underset{y_i}{\text{argmax}}\, \pi_\theta(y_i|x) \text{ for } y_i \in y\}. \quad (2)$$

We train our model using a policy gradient technique (Sutton et al., 1999). Unlike typical sequential reinforcement learning scenarios, our π_θ only performs one action for a given input, receiving the corresponding reward immediately, without transitioning through other intermediate states. Therefore, our setup is similar to a contextual multi-armed bandit problem (Langford and Zhang, 2008), where each "arm" corresponds to a particular label sequence $y = (y_0, y_1, ..., y_n) \in \{0, 1\}^n$. However, in our scenario, the policy is generally allowed to access rewards for multiple possible actions via sampling, which is different from typical bandit settings where only one $(action, reward)$ pair is available for each episode.

The training objective is to maximize the expected reward assigned to a predicted label sequence y for a given input x, computed by the reward function R:

$$J(\theta) = \mathbb{E}[R(x, y)] \quad (3)$$

The policy gradient theorem states that the gradient of this expectation can be expressed as follows (Sutton et al., 1999):

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}[R(x, y) \log \pi_\theta(y|x)] \quad (4)$$

Since the above expectation is intractable for a large dataset and the corresponding action space, this gradient is estimated by sampling:

$$\nabla_\theta J(\theta) = \nabla_\theta r^s \log \pi_\theta(y^s|x), \quad (5)$$

where $y^s \sim \pi_\theta(*|x)$ is a sample from the current policy at a given step, consisting of binary token labels $y^s = (y^s_0, y^s_1, ..., y^s_n)$, and $r^s = R(x, y^s)$.

As is commonly done when using policy gradients, we subtract a baseline from the reward for variance reduction. We instantiate the baseline as $r^a = R(x, y^a)$, the reward given to the the most likely label sequence y^a according to the current policy. The gradient becomes:

$$\nabla_\theta J(\theta) = \nabla_\theta (r^s - r^a) \log \pi_\theta(y^s|x) \quad (6)$$

Accordingly, we train our model by minimizing the following loss function:

$$\ell_\theta = (r^a - r^s) \log \pi_\theta(y^s|x). \qquad (7)$$

Using the baseline r^a allows the intuitive interpretation that a sample y^s is encouraged if its reward is higher than the current policy's prediction, i.e., when factor $(r^a - r^s)$ is negative, and discouraged otherwise.

Best-of-k Sampling

Prior work with a similar application of policy gradient (Dong et al., 2018; Laban et al., 2021) observed an advantage in sampling k times and taking the average loss over all samples rather than using a single sample. However, in our experiments, we observe that only using the sample with the *maximum* reward from a large number of samples works significantly better than taking the average or only sampling once. A large k improves the discovery of high-quality compressions – if we only use a single sample or a very small k, we observe a higher tendency of models to converge on simple behaviors with low reward improvements, such as only extracting the first-L tokens of a sentence. The choice of k controls a trade-off: with a higher k, we spend more time computing the rewards of samples and less on model updates, given a limited wall-time constraint for training. We determine k in an unsupervised manner using a validation set (details in Section 5.2).

4.2 Model Architecture

π_θ is initialized as a transformer encoder model with a linear classification head. In particular, we use the 6-layer DistilRoBERTa model (Sanh et al., 2019) due to its efficiency and smaller size compared to other BERT-like models, while retaining good results on the GLUE benchmark[1]. During training, the whole model is fine-tuned. For each token in the input, our model will determine whether it should be kept or filtered. Figure 2 visualizes the design. This architecture produces summaries in an instantaneous, non-autoregressive fashion, allowing for fast prediction (see Section 5.6).

4.3 Reward Functions

We do not have direct access to ground-truth training data in our setup, so we consider a suite of reward functions that may correlate with different aspects of sentence compression quality.

Fluency

This reward function is intended to ensure grammatically correct and well-written sentences. We use a masked language model (LM) to estimate the fluency of a compressed sentence. In particular, we compute fluency as the average logit of a token y_i in the compressed sentence y. We do this without masking y_i to reduce the running time during training, as masking would require to re-encode the sentence for each token. Based on our experiments, this simplification still produces good estimates of fluency.

$$R_{fluency}(y) = \frac{1}{|y|} \sum_{i=1}^{|y|} \text{LM}(y_i|y) \qquad (8)$$

We normalize $R_{fluency}$ by dividing it by an empirically set constant, to keep its values in a similar range compared to the other rewards. The constant is an observed minimum value from a sample dataset. We argue that a masked language model is more appropriate in our setup compared to a left-to-right (causal) language model – when predicting or sampling a compressed sentence during training, the sentence is treated as a finished rather than an intermediate output, which is not captured by the auto-regressive inference of causal LMs. We confirm the advantage of a masked LM over a left-to-right LM in a comparison on a development set (Appendix A).

We note the precedent for using language models to measure fluency: Zhou and Rush (2019) and Schumann et al. (2020) use language models trained on a summarization target domain, e.g., headlines. Laban et al. (2020) uses a generic causal language model to estimate fluency. Niu et al. (2019) use a masked language model to score candidate compressions.

Similarity-to-Source

The similarity reward is intended to preserve the meaning of the source sentence in the compressed sentence. We experiment with several options to compute similarity, all using models from the `sentence-transformers` library[2] (Reimers and Gurevych, 2019):

- **Bi-Encoder Similarity**: A sentence encoder f separately computes embeddings for the source and the predicted summary. We calculate the cosine similarity between both embeddings: $R_{sim}(x,y) = cos(f(x), f(y))$

---

[1] https://huggingface.co/distilroberta-base

[2] https://www.sbert.net/

- **Cross-Encoder Similarity**: Output of a cross-encoder model f_{sim} measuring the semantic textual similarity between both sentences: $R_{sim}(x, y) = f_{sim}(x, y)$

- **Cross-Encoder NLI:** We also test a natural language inference (NLI) model f_{nli} to estimate how well a compressed sentence retains source information. The intuition is that the source should *imply* information in the output: $R_{nli}(x, y) = f_{nli}(y|x)$

Based on experiments on a development dataset, the bi-encoder similarity performs best in our setup.

Length and Compression Ratio

Because our model is non-sequential, we cannot easily employ a hard constraint to control the length of compressed sentences. Instead, we impose a soft length control using Gaussian reward functions. In particular, we either use a reward function for the length (token count) in a compressed sentence R_{len}, or one for the compression ratio between the source and prediction, in terms of token counts, R_{cr}. We choose one of these two depending on whether a consistent length or a consistent ratio is desired, which differs for different evaluation datasets. We set the distribution means of both rewards as the desired values for word count and compression ratio. We set the standard deviations as the mean times a factor s which we set to 0.4 for both reward functions (Equations 9, 10):

$$R_{len} = \mathcal{N}(\mu_L, (s \times \mu_L)^2), \qquad (9)$$

$$R_{cr} = \mathcal{N}(\mu_{cr}, (s \times \mu_{cr})^2). \qquad (10)$$

Reward Aggregation

The final reward function is an average of the reward functions $R_{fluency}$, R_{sim}, combined with either R_{len} or R_{cr}:

$$r_{total}(x, y) = \frac{1}{3} \sum_i R_i(x, y). \qquad (11)$$

In practice, when the downstream task is known, reward functions may be designed and calibrated based upon insights and domain expertise, e.g., an optimal summary length for a specific application or different language models corresponding to different summary styles. In this work, we only use publicly available and commonly-used off-the-shelf models to construct reward functions.

5 Experiments

This section presents a detailed analysis and evaluation results for our proposed model. We name our model **SCRL (Sentence Compression with Reinforcement Learning)**. We make all code, model outputs and data available[3].

5.1 Datasets

5.1.1 Training Datasets

We use two datasets for training: Newsroom (Grusky et al., 2018) and Gigaword (Rush et al., 2015). For Newsroom, we extract the first three sentences from each article, only keeping sentences with a number of tokens between 15 and 60. Newsroom was chosen due to the large size and a variety of un-preprocessed news articles from different sources. Ground-truth summaries are not included in the training data, thus the two datasets are treated as large unlabeled text collections. We train a model for short headline-like summaries on Gigaword to evaluate it on the Gigaword test set, which comes in a specific preprocessed format[4]. Training on Gigaword allows to expose the model to the same preprocessing, for a fair evaluation.

5.1.2 Development Dataset

We constructed a small labelled validation dataset for model development: we automatically identified sentence-summary pairs in Newsroom, also including title-summary pairs, by extracting cases where the tokenized summary is contained in a tokenized sentence, with preserved order. We manually filter a subset of these examples based on grammaticality and informativeness and obtain 280 examples. This dataset was only used during initial development to compare the different reward function variants discussed in Section 4.3.

5.1.3 Evaluation Datasets

The evaluation includes five test sets – key statistics are listed in Table 1. L_{src}, L_{tgt} are the token counts in source and target sentences and $cr = L_{tgt}/L_{src}$ is the compression ratio. Following Schumann et al. (2020), we compare our models on Gigaword against baselines of comparable length brackets using ROUGE F1-scores[5]. For DUC2004 (Task

Testset	Type	Size	L_{src}	L_{tgt}	cr
Gigaword	abs	1951	29.7	8.8	0.4
DUC2004	abs	500	32.9	11.9	0.41
Google	ext	1000	27	11	0.45
Broadcast	ext	1370	19.8	14.4	0.76
BNC	ext	1629	27.9	19.3	0.72

Table 1: Overview of the evaluation datasets. The *Type* column indicates whether the ground-truth is extractive or abstractive. *Size* gives the number of sentences.

Name	Train data	Test Data	Time
SCRL-L8	Gigaword	Gigaword	9
SCRL-L11	Newsroom	DUC04, Google	9.5
SCRL-CR75	Newsroom	Broadcast, BNC	10

Table 2: Overview of trained models and training time in hours.

1), following prior work, we truncate model outputs to 75 characters and compute ROUGE recall scores. While Gigaword and DUC2004 contain abstractive ground-truth summaries, the remaining three datasets have token-level extractive ground-truth summaries. The ground-truth compressions in the Google sentence compression dataset (Filippova and Altun, 2013) were automatically generated using grammatical constraints and distant supervision via headlines. The Broadcast and BNC datasets (Clarke and Lapata, 2008) contain manually created extractive sentence compressions which tend to be longer compared to the other evaluation datasets. Following previous work, we report a simple F1-score based on tokenized predicted and ground-truth summaries on the three extractive datasets, but also measure ROUGE F1 scores.

5.2 Model Development

We tune our approach in several phases. At first, we identify an optimal learning rate and batch size using a grid search with a fixed training duration. We compare different settings based on the average reward achieved on a unlabelled, held-out set of the training data. Next, we test different values of k (1, 5, 10, 50, 100), the number of samples per step, and pick the best k based on the average reward on the validation set. This method of hyperparameter tuning is fully unsupervised.

Using learning rate $1e-05$, batch size 4 and $k = 100$ identified in the previous runs, we next compare the different options for the similarity reward listed in Section 4.3 and pick the best (bi-encoder similarity) based on the F1-score on our labelled Newsroom-based validation set (see Appendix B).

5.3 Training

We initialize the encoder component of our model with the pretrained 6-layer DistilRoBERTa model (Sanh et al., 2019). The binary classifier module

is initialized randomly. We train each model for 8,000 steps with a batch size of 4 on a Google Cloud virtual machine with one NVIDIA Tesla T4 GPU, using the AdamW optimizer (Loshchilov and Hutter, 2019). Our default reward combination contains masked-LM fluency and bi-encoder similarity combined with either R_{len} or R_{cr}. Table 2 gives an overview of the three models that are used in the evaluation. Note that the sample size of 100 is responsible for the long training durations. SCRL-L8 and SCRL-L11 are trained with R_{len} whereas SCRL-CR75 is trained with R_{cr}, with a compression ratio of 0.75. This is because the ground-truth summary lengths are approximated better by a fixed length rather than a fixed ratio in the Google and DUC2004 datasets, whereas a fixed ratio describes the Broadcast and BNC datasets better.

5.4 Baselines

We compare our model to the greedy stochastic hill climbing approach in Schumann et al. (2020) which obtained state-of-the-art ROUGE results for unsupervised baselines on the Gigaword and DUC2004 datasets. Because this method and SCRL do not have identical objective functions, we implement the hill climbing algorithm applied to our reward functions, which we will name HC throughout this work. This allows for a clearer comparison between RL and discrete search. HC optimizes $R_{fluency}$, R_{sim} under fixed length constraints instead of using R_{len} and R_{cr}. Different from Schumann et al. (2020), it runs for a fixed number of 2000 steps and restarts only when the search is stuck rather than in equal intervals (details in Appendix E). We analyze the performance of HC for different budgets to understand at what point search can surpass the learned policies. We also compare against Zhou and Rush (2019), Niu et al. (2019) and the RL-based method by Wang et al. (2018) on datasets where results are available.

5.5 Evaluation Results

Table 3 shows the evaluation results on all used test datasets. Results of methods apart from SCRL and HC are taken from previous works. We com-

Dataset	Model	ROUGE			F1	L_d	L_o	cr_o	Inf. Time (s)
		1	2	L					
Gigaword	Lead-L8	21.39	7.42	20.03		8	7.9		
	Zhou and Rush (2019)	26.48	10.05	24.41			9.3		
	Schumann et al. (2020) L8	26.32	9.36	24.19		8	7.9		
	Schumann et al. (2020) L10	28.80	**10.66**	25.82		10	9.8		
	HC-L8	28.00	8.53	25.90		8	7.96	0.31	11.733
	SCRL-L8	**29.64**	9.98	**26.57**		8	7.68	0.28	**0.004**
DUC2004	Zajic et al. (2004)[★]	25.12	6.46	20.12					
	Baziotis et al. (2019)	22.13	6.18	19.3					
	West et al. (2019)	22.85	5.71	19.87					
	Schumann et al. (2020)	**27.41**	**8.76**	23.89		13			
	HC-L11	27.40	8.65	**24.16**		11	10.69	0.36	12.305
	SCRL-L11	25.27	7.82	22.14		11	10.58	0.35	**0.004**
Google	Filipova[★]				0.82			0.38	
	Wang et al. (2017)[★]				0.8			0.43	
	Wang et al. (2018)				0.565				
	Zhou and Rush (2019)				0.61				
	Niu et al. (2019)				0.5				
	HC-L11	68.04	49.21	67.40	0.637	11	11.0	0.46	11.261
	SCRL-L11	**70.22**	**53.03**	**69.84**	**0.711**	11	10.8	0.44	**0.004**
Broadcast	Wang et al. (2017)[★]				0.66				
	Wang et al. (2018)				0.665				
	HC-CR75	82.20	63.78	81.76	**0.792**	75%	14.9	0.77	13.516
	SCRL-CR75	**82.22**	**66.01**	**81.78**	0.787	75%	15.1	0.78	**0.004**
BNC	Wang et al. (2017)[★]				0.66			0.53	
	Wang et al. (2018)				0.675				
	HC-CR75	78.91	60.10	78.13	**0.768**	75%	21.0	0.76	15.268
	SCRL-CR75	**79.49**	**62.32**	**78.63**	0.765	75%	21.0	0.76	**0.004**

Table 3: Results on evaluation datasets. [★] indicates supervised models. ROUGE F1-scores are shown for all dataset but DUC2004, where ROUGE recall is used. L_d: desired output length, L_o / cr_o: actual average length / compression ratio of the outputs.

pute ROUGE scores using the implementation from Google Research[6]. On Gigaword, SCRL outperforms all baselines, except Schumann et al. (2020) with a 10 token constraint in ROUGE-2. On DUC2004, SCRL remains behind the hill climbing methods, but outperforms other unsupervised baselines. On the Google dataset, SCRL obtains state-of-the-art results among unsupervised methods. On Broadcast and BNC, SCRL and HC obtain very similar scores, which are both higher than previously reported results. Figure 3 shows ROUGE-1 scores obtained by HC at different search budgets, compared to SCRL. The hill climbing strategy approaches or outperforms the trained model at different paces, depending on the dataset.

Interestingly, HC still achieves higher rewards than SCRL relatively early during its search (see Appendix F), which is inconsistent with the evaluation results. Potential reasons for this disparity are disadvantages through the hard length constraints, a mismatch between the heuristic reward functions and evaluation metrics, and beneficial biases induced through our training framework.
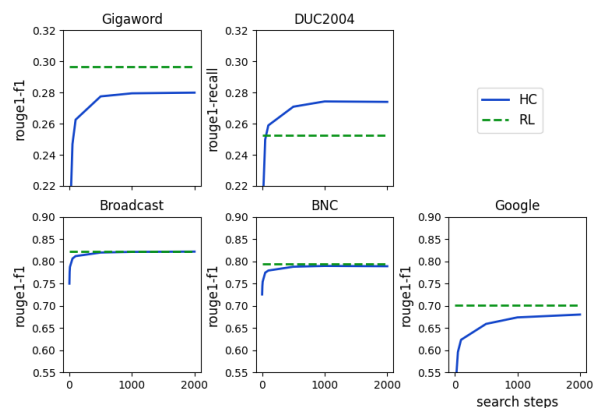
Figure 3: Evaluation scores of RL model compared to hill climbing algorithm (HC) at different search budgets.

5.6 Prediction Running Times

We compare the inference-time speed of SCRL with HC using different budgets of search steps[7]. The fastest batch size for both approaches is used. The *Inference Time* in Table 3 shows the average number of seconds per processed sentence, with the number of search steps set to $T = 2000$ for HC. SCRL is roughly $4000\times$ faster than HC with $T =$

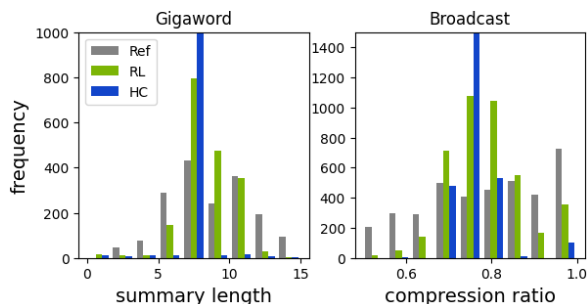[7]On a Google Colab Notebook with a Tesla P-100 GPU

Figure 4: Distribution of summary lengths and compression ratios. The maximum frequencies of *HC* are cropped.
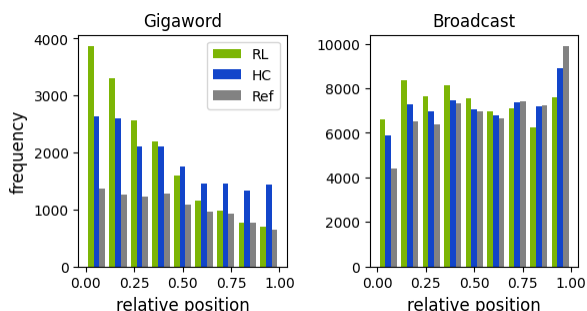

Figure 5: Distribution of relative positions from which tokens are extracted in source sentences.

2000, and $\sim 200\times$ faster when T is reduced to 100, for example. We believe that such a speed-up with a preserved evaluation performance is a critical factor when considering real-world applications of sentence compression.

6 Analysis

6.1 Summary Length and Extraction Regions

The length and compression ratio of summaries produced by SCRL is distributed around the desired values, with peakier distributions than in ground-truth summaries (examples in Figure 4). HC produces exactly the desired value whenever possible, due to the enforced constraint for length or ratio. Figure 5 shows how SCRL and HC extract tokens from different relative positions within source sentences. SCRL has a higher tendency to extract early tokens. We hypothesize that this is a reliable high-reward strategy discovered during training, considering that a milder form of the lead-bias also shows in HC. Note that neither method is inherently biased in its design to prefer tokens from certain regions.
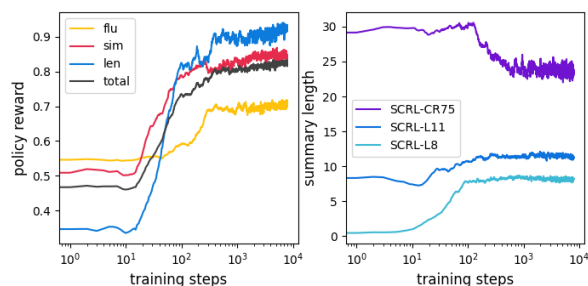

Figure 6: Development of reward functions (of SCRL-L11) and summary length during training (all models), from a moving average over 50 steps, using a log scale.

6.2 Training Dynamics

Figure 6 shows how rewards and summary length develop throughout training. The rewards generally increase quickly in the first few hundred training steps and then continue to grow very slowly. Fluency starts to increase later than the other reward functions, which is likely related to our observation that it is more sensitive to small changes in a summary. Interestingly, the summary lengths develop differently depending on the length or compression setting – SCRL-L8 and SCRL-L11 start with short summaries and increase the size over time whereas SCRL-CR75 starts with long summaries before settling on a shorter certain range.

6.3 Learned Summarization Techniques

Our models learn a variety of behaviors to compress sentences, such as removing articles, auxiliary verbs, relative clauses and temporal expressions. Figure 7 shows some examples.

6.4 Error Analysis

Even though our models learn to produce grammatical sentences fairly well, grammatical errors do still appear, and are more common for the models with a short output length (SCRL-8, SCRL-11). In some cases, semantic errors occur where the original meaning is changed or made unintelligible. Both SCRL and HC are susceptible of semantic and grammatical errors, as can be seen in some examples in Appendix G. A type of error that is specific to SCRL is the splitting or merging of tokens resulting from its operation on Byte Pair Encoding-based subword tokens (more details in Appendix C).

6.5 Customization via Reward Functions

To demonstrate that our approach is flexible for customization, we pick a simple example of re-

Figure 7: Examples of learned summarization techniques. Selected tokens are marked in green and common removal behaviors are pointed out with underlining.

programming model behavior using a hand-crafted reward function. We note that in some cases, the model unnecessarily keeps day references in compressed sentences, such as "Thursday" or "yesterday". We construct a simple reward function that returns zero if any day-like word from a small gazetteer appears in an output and a score of 1 otherwise. We fine-tune an existing model with this additional reward and observe that it successfully avoids including day-words that the previous model would include. Importantly, it additionally learned to remove other tokens attached to day-words, e.g. "on" in "on Monday", keeping the sentences grammatical. Table 4 shows some examples. Empirically, the new model's outputs contain words from the gazeteer in 1% of cases where they appear in the source, compared to 12% in the initial model.

Initial Model	Customized Reward
The burrito chain said **on Tuesday** that comparable sales fell 26.1% last month.	The burrito chain said that comparable sales fell 26.1%.
His car was found **last Thursday** alongside Rubyvale Road.	His car was found alongside Rubyvale Road.

Table 4: Example outputs of model with customized reward function to exclude mentions of days.

7 Discussion

We argue that RL offers the following advantages over discrete search strategies for sentence compression and similar text editing or generation tasks. The necessary search and exploration is moved into the training stage, allowing fast inference independently of how efficient objectives are to compute. Furthermore, discrete search unnecessarily spends time navigating through low-quality outputs that a trained model can quickly learn to avoid. Limitations of our approach compared to the search-

based approach are its lesser flexibility in terms of on-the-fly customization and a sensitivity to disparities between training data and the application domain. Furthermore, the trained models show a lower capability to optimize the selected objectives compared to search, though this does not have a negative impact on the evaluation in most cases.

The fact that most of our training time is spent on estimating the quality of sampled compressions due to large sample size k, shows that our approach is somewhat similar to large-scale search strategies applied to a whole dataset, with the difference that the sampling behavior at each step changes over time and is informed by previous steps. This suggests that discrete search could support the RL training, similarly to the learning-from-search approach described by (Li et al., 2020).

8 Conclusion

This work presents a simple and effective approach for learning sentence compression models based on objective functions rather than ground-truth examples. Because it is unsupervised, it is well-suited for creating customized applications even when no gold training data is available, allowing for task-specific tuning based on arbitrary sets of reward functions, which do not need to be differentiable. Importantly, our approach is very fast at inference time compared to alternative discrete search-based methods. We are interested in several future directions related to this work: 1) systematic approaches to design reward functions for summarization, 2) RL-based summarization models with length control on the fly, 3) testing our approach on other languages, and 4) the design of curricula for different reward functions as they might pose varying difficulties at different stages of the training.

References

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. Seq^3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression.

Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. 2019. Better rewards yield better summaries: Learning to summarise without references. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3110–3120, Hong Kong, China. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384, Sydney, Australia. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium. Association for Computational Linguistics.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8.

Thibault Févry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. 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 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Chiori Hori and Sadaoki Furui. 2004. Speech summarization: An approach through word extraction and a method for evaluation. *IEICE Trans. Inf. Syst.*, 87-D(1):15–25.

Philippe Laban, Andrew Hsi, John Canny, and Marti A Hearst. 2020. The summary loop: Learning to write abstractive summaries without examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5135–5150.

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6365–6378, Online. Association for Computational Linguistics.

John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Jingjing Li, Zichao Li, Lili Mou, Xin Jiang, Michael Lyu, and Irwin King. 2020. Unsupervised text generation by learning from search. *Advances in Neural Information Processing Systems*, 33:10820–10831.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Chanakya Malireddy, Tirth Maniar, and Manish Shrivastava. 2020. SCAR: Sentence compression using autoencoders for reconstruction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 88–94, Online. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. 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 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.

Tong Niu, Caiming Xiong, and Richard Socher. 2019. Deleter: Leveraging bert to perform unsupervised successive text compression. *arXiv preprint arXiv:1909.03223*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Raphael Schumann, Lili Mou, Yao Lu, Olga Vechtomova, and Katja Markert. 2020. Discrete optimization for unsupervised sentence summarization with word-level extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5032–5042.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3246–3256, Hong Kong, China. Association for Computational Linguistics.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674, Melbourne, Australia. Association for Computational Linguistics.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL – International Journal of Applied Linguistics*, 165:259–298.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2020. Learning to summarize from human feedback. *arXiv preprint arXiv:2009.01325*.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1057–1063. The MIT Press.

Vincent Vandeghinste and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Text Summarization Branches Out*, pages 89–95, Barcelona, Spain. Association for Computational Linguistics.

Liangguo Wang, Jing Jiang, Hai Leong Chieu, Chen Hui Ong, Dandan Song, and Lejian Liao. 2017. Can syntax help? improving an LSTM-based sentence compression model for new domains. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1385–1393, Vancouver, Canada. Association for Computational Linguistics.

Liangguo Wang, Jing Jiang, and Lejian Liao. 2018. Sentence compression with reinforcement learning. In *International Conference on Knowledge Science, Engineering and Management*, pages 3–15. Springer.

Peter West, Ari Holtzman, Jan Buys, and Yejin Choi. 2019. BottleSum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3752–3761, Hong Kong, China. Association for Computational Linguistics.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119.

David M Zajic, Bonnie J Dorr, and Jimmy Lin. 2008. Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4):1600–1610.

Yang Zhao, Zhiyuan Luo, and Akiko Aizawa. 2018. A language model based evaluator for sentence compression. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 170–175.

Jiawei Zhou and Alexander M Rush. 2019. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5101–5106.

A   Masked vs. Causal Language Model for Fluency

We compare the masked DistilRoBERTa[8] language model to the causal DistilGPT2[9] model on our development dataset. Both models have roughly the same number of parameters (82M). Table 5 shows the results.

LM	DistilRoBERTa	DistilGPT2
F1-Score	0.565	0.546

Table 5: Comparison of a masked and a causal language model to estimate fluency.

B   Similarity Functions

Table 6 compares different variants of the similarity reward functions on our development dataset.

Similarity	Bi-Sim	Cross-Sim	NLI
F1-Score	0.624	0.598	0.564

Table 6: Comparison of similarity reward functions on our development dataset.

C   Error Analysis: Split and Merged Tokens

One type of error is the splitting or merging of tokens from the source which results from the fact our model predicts labels at the level of BPE subword tokenization used in the pretrained language model that we finetune. While some of these occurrences are minor, e.g. 'St.' → 'St', or even useful compressions, e.g. '29th' → '29', many of these cases produce noisy outputs, e.g 'Perigord' → 'Perig'. Based upon analysis of prediction behavior, we estimate that 6% of output sentences contain some form of this phenomenon.

D   Implementation Details

D.1   Pretrained Model IDs

Table 7 lists the model IDs of all open-source pretrained models used in this work, which can be found at https://huggingface.co.

Algorithm 1 Stochastic First-Choice Hill Climbing

input objective function $R(x, y)$, source sentence x, summary length L, number of steps T, initialization function $Init(x, L)$, sampling function $S(y)$
$\quad y^0 \leftarrow Init(x, L)$
$\quad$**for** $t = 1$ to T **do**
$\quad\quad y' \leftarrow S(y^{t-1})$
$\quad\quad$**if** $R(x, y') \geq R(x, y^{t-1})$ **then**
$\quad\quad\quad y^t = y'$
$\quad\quad$**else**
$\quad\quad\quad y^t = y^{t-1}$
$\quad$**return** y_t

D.2   Tokenization

We use the NLTK[10] Punkt Tokenizer for several purposes in this work:

- Obtaining the sentence length and compression ratio in R_{len} and R_{cr}.

- Compressing sentences by selecting tokens in our hill climbing implementation HC.

- Obtaining source and summary tokens to compute the F1-score in Table 3, except for source and reference tokens on the Gigaword, Broadcast and BNC datasets which are pretokenized.

The involved transformer models (SCRL, $R_{fluency}$, R_{sim}) internally tokenize sentences based on Byte Pair Encoding.

E   Hill Climbing Baseline

Algorithm 1 shows the hill climbing search approach HC used in our experiments, which is based on Schumann et al. (2020). At the beginning, a binary label sequence y^0 is initialized by setting L randomly selected labels to 1 and the rest to 0. At each step t, $S(y)$ samples a new label sequence y' by randomly selecting a positive and a negative-valued label y_i and y_j and swapping their value. Note that this always keeps the number of tokens at L. The sampled y' is accepted if it obtains a higher or equal objective score than the previously best candidate y^{t-1}. We keep track of previously created sequences and skip these. If no new label sequence can be discovered at step t, we terminate the algorithm and restart it with $T - t$ remaining steps. In the end, the highest-scoring y found across different runs is returned. We generally set T to 2000 and keep track of intermediate results to evaluate HC also at fewer search steps. Due to this,

---

Model Usage	Model ID
Encoder initialization for SCRL models	`distilroberta-base`
Masked LM Fluency Reward	`distilroberta-base`
Causal LM Fluency Reward	`distilgpt2`
Bi-Encoder Similarity Reward	`all-distilroberta-v1`
Cross-Encoder Similarity Reward	`cross-encoder/stsb-distilroberta-base`
Cross-Encoder NLI Reward	`cross-encoder/nli-distilroberta-base`

Table 7: Overview of pretrained models used throughout this work.

we decided restart the search dynamically rather in equal-paced intervals, which we believe should be tuned with respect to a known maximum budget T.

F   Rewards Obtained by HC vs. SCRL

Figure 8 compares the $R_{fluency}$ and R_{sim} rewards of SCRL to HC with different search budgets. The length and compression ratio rewards are not included as these are enforced through a constraint by HC. Note that these figures need to be interpreted carefully as they assume that both approaches produce summaries of comparable lengths. For example, the similarity reward tends to increase with the summary length.

G   Output Examples

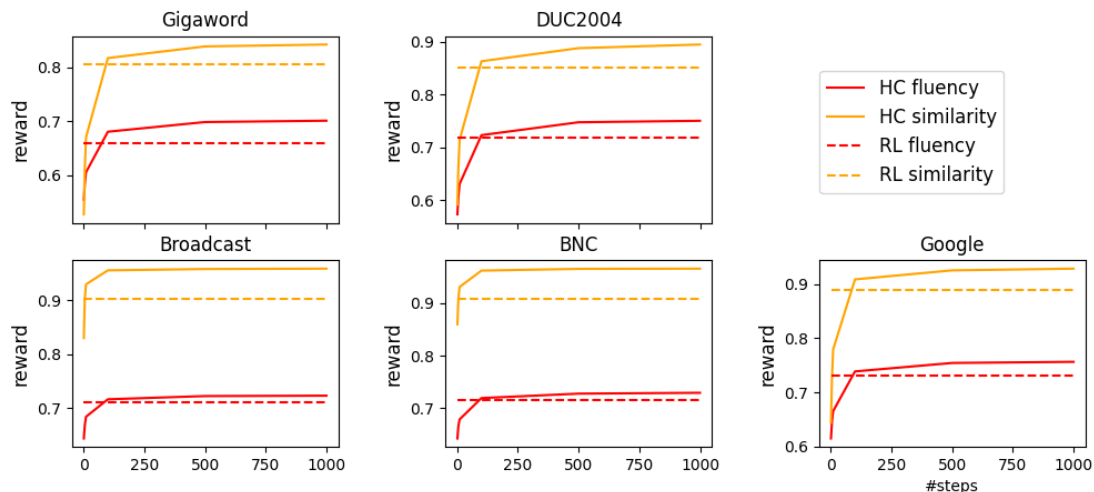Table 8 lists a few examples outputs produced by SCRL and HC.

Figure 8: Rewards of RL model compared to hill climbing algorithm (HC) at different search budgets.

Source	the us space shuttle atlantis separated from the orbiting russian mir space station early saturday , after three days of test runs for life in a future space facility , nasa announced .
SCRL-L8	the space shuttle atlantis separated from russian station
HC-L8	atlantis space station after test runs for nasa
Source	a katyusha rocket fired from lebanon on saturday morning hit the western galilee in north israel , causing two lightly hurt , israel radio reported .
SCRL-L8	katyusha rocket fired from lebanon hit galilee israel
HC-L8	katyusha rocket fired hit western galilee israel israel
Source	Manchester United have agreed a £35m deal to sign Sporting Lisbon midfielder William Carvalho, according to talkSPORT.
SCRL-L11	Manchester United agreed £35m deal to sign Lisbon midfielder William Carvalho.
HC-L11	Manchester United have agreed a £35m deal to sign William Carvalho
Source	Egyptian President Hosni Mubarak met here Sunday with Syrian President Hafez Assad to try to defuse growing tension between Syria and Turkey.
SRCL-L11	Egyptian President Hosni Mubarak met with Syrian President Hafez Assad def.
HC-L11	Egyptian President Hosni Mubarak met Sunday with Syrian President Hafez Assad
Source	Russian President Boris Yeltsin, who is still recuperating from his latest illness, has canceled a trip to an Asian summit next month, his office said Friday.
SCRL-L11	Russian President Boris Yeltsin recuperating has canceled a trip to Asian summit.
HC-L11	Russian President Boris Yeltsin has canceled trip to an Asian summit
Source	Laurie had a passion and a warmth for people rather than the state .
SCRL-CR75	Laurie had a passion and warmth for people.
HC-CR75	Laurie had a passion and warmth for the state .
Source	And speaking of the royals , the Duchess of York , Sarah Ferguson , was in Los Angeles last week holed up at the Four Seasons Hotel and when she ventured out , I hear she visited some of the studios like Sony to have meetings involving TV projects .
SCRL-CR75	And speaking of the royals, the Duchess of York, Sarah Ferguson, was in Los Angeles last week holed up at the Four Seasons Hotel and I hear she visited studios like Sony to have meetings.
HC-CR75	And speaking of royals , Duchess of York Sarah Ferguson was in Los Angeles last week at the Four Seasons Hotel and when she ventured out she visited some of the studios like Sony to have meetings .
Source	Of the 24,058 people interviewed , 37.7 per cent of women attended arts events and 33.1 per cent of men .
SCRL-CR75	Of 24,058 people interviewed, 37.7 per cent of women attended arts events and 33.1.
HC-CR75	Of 24,058 people interviewed , 37.7 per cent women attended arts events and 33.1 men .

Table 8: Output examples, with semantic and grammatical errors highlighted.