Salience Estimation and Faithful Generation

Modeling Methods for Text Summarization and Generation

Thesis Proposal

Chris Kedzie
Department of Computer Science
Columbia University
kedzie@cs.columbia.edu

November 28th, 2018

Abstract

Automatic text summarization is one of the longest-standing application areas in the field of natural language processing (NLP), with a history as deep as some of the more foundational tasks, e.g. syntactic parsing (Yngve, 1955) or part-of-speech tagging (Harris, 1962). While there are many variants, the general summarization task is to reduce a large input text into its most essential pieces of information, and in doing so reduce the amount of reading a human has to do. In this thesis we focus on advances to two summarization subtasks: *(i)* identifying the most import content for inclusion in the summary, and *(ii)* rendering that content in such a way as to not misrepresent the original input. We refer to the former as ***salience estimation*** and the latter as ***faithful generation***.

With respect to problem *(i)* we propose two novel methods for working in low context, streaming news scenarios using feature-based regression, clustering, and learning-to-search. Additionally, we develop several hierarchical models of sentence salience using deep neural networks. We perform analyses of different neural architecture choices in the context of single document summarization across multiple genres along with ablation studies to understand what signals in the data are most important for model training. Based on these experiments, we find impediments to learning useful lexical features at the sentence level, and propose a novel word importance model with frequency and surprisal based feature embeddings to overcome these limitations. We also propose a method of domain adaptation of this word importance model from single-document summarization to multi-document summarization.

Finally, we introduce a novel framework for generating text that is faithful, i.e. respects prior knowledge or input data, in an effort to provide stronger guarantees about summary reliability and prevent the hallucination of facts. In particular, we model generation as a two player game between a *generator*, a conditional language model that produces a text utterance given an input (either text or table data), and a *recognizer*, a classifier that tries to accurately predict the original input data from the generator's utterances. We explore this framework in both data-to-text and text-to-text settings. In the data-to-text setting, the recognizer tries to predict the values of fields from the conditioning table data, e.g. predicting someone's occupation from a brief biographical description produced by the generator. In the text-to-text setting, the recognizer answers cloze style questions (Taylor, 1953) about the input text, given an abstractive summary produced by the generator. The generator uses the recognizer as a learning signal to perform beam search optimization (Wiseman and Rush, 2016); the goal here is to learn a generator that is capable of producing syntactically diverse utterances that respect the conditioning input data. We additionally discuss how this framework can be used as an alternative method of controllable text generation, with the added benefit that we learn a recognizer model that can give confidence scores about the faithfulness of a given utterance.

Contents

1 Introduction

Everyday we ask friends and colleagues to do it, to tell us about books, newspaper articles, and other complex texts, and, without more than a moment's reflection, they are able to compress their experience into succinct natural language statements that describe the subject of our request. The ease with which we summarize belies the difficulty of writing programs that do so, and so it would appear that the robust ability to create summaries is, as of 2018, a uniquely human capability. This partially explains summarization's allure to the artificial intelligence (AI) research community, which has in one way or another, attempted to aggregate and naturally compress text data since at least the 1950's (Luhn, 1958). This makes automatic text summarization one of the longest-standing application areas in the field of natural language processing (NLP), with a history as deep as some of the more foundational tasks, e.g. syntactic parsing (Yngve, 1955) or part-of-speech tagging (Harris, 1962).

While there are many variants, the general summarization task is to reduce a large input text into its most essential pieces of information, and in doing so reduce the amount of reading a human has to do. In this thesis we focus on advances to two summarization subtasks: *(i)* identifying the most import content for inclusion in the summary, and *(ii)* rendering that content in such a way as to not misrepresent the original input. We refer to the former as ***salience estimation*** and the latter as ***faithful generation***.

Salience Estimation Our completed and proposed methods of salience estimation cover a range of techniques including regression, clustering, learning-to-search, and deep neural networks. We experimentally verify the utility of our approaches across a variety of summarization tasks including stream summarization, single-document summarization, and multi-document summarization. The general research goal here is to develop flexible models for identifying important words, phrases, and sentences that are likely to serve as representative members of the larger text in which they are found.

In extractive summarization, where the summary is constructed by copying and pasting phrases or sentences from the input text, salience estimation can be used directly to create a summary. For example, a simple method of sentence extractive summarization would be to order the input sentences by decreasing salience and select the top few sentences for the summary. In most cases, salience is not the only consideration for summary inclusion; redundancy, discourse coherence, fluency, and many other metrics of text quality can and have been used as content selection criteria. These measures are largely independent of salience (with the notable exception of redundancy) and so we do not explore them in detail here. The details of our completed and proposed work on salience estimation can be found in sections 2 and 3.

In section 2 we develop two feature-based models of sentence salience and evaluate them in a relatively novel stream summarization crisis-monitoring scenario (Starbird and Palen, 2013; Aslam et al., 2015, 2016). In this task, a user is interested in tracking information about a disaster event (e.g. a hurricane). The user provides a text based query describing the event (e.g. "Hurricane Sandy") to the summarization model. The model must then monitor a stream of news articles, extracting important sentences in the document stream and present them to the user. The information must be highly salient to the query event, as well as novel and timely to the user – we do not want to bombard the user with irrelevant or repeated information, and our methods cannot take so long that the extracted information is out of date.

We develop and compare several deep learning based models of word and sentence salience in section 3 and evaluate them primarily on sentence extractive single document summarization (SDS). We perform this evaluation across a variety of genres including news, personal narratives, and medical journal articles. Intuitively, the goal here is to summarize a single news article, short story, or research paper by selecting a subset of the input sentences to serve as the summary. We also explore adaptation of these algorithms to the sentence extractive multi-document summarization (MDS) task, where there is much less training data. In this task, an extractive summarization model is given a set of related documents (typically ten), and must select a subset of the sentences to use as the summary of the document set.

Faithful Generation Finally, section 4 describes our proposed contributions to abstractive summarization, i.e. methods that produce novel summary text using a generative model. Research in abstractive summarization is increasing, in part due to the success of general purpose sequence-to-sequence transduction models for machine translation (MT) that have been ported to the summarization task. The ability of abstractive models to generate fluent summaries is impressive. However, they are also especially prone to generating generic statements and hallucinated facts that are not grounded by evidence in the input. We are interested in this problem from both the perspectives of selecting the right evidence in the input for generation, a task for which we enlist our salience estimation methods, and also certifying that a generated text conforms to knowledge represented in the input or possessed by the model *a priori*. For the latter goal, we propose to model generation as a two player game, where one player, the generator, is tasked with producing a text utterance describing a piece of evidence (either text or table data); the second player, the recognizer, evaluates the plausibility of the utterance with respect to the evidence. This overall regime will constitute our proposed method called faithful generation. We propose evaluation both on an abstractive SDS task (Völske et al., 2017), as well as, two data-to-text generation test beds (Lebret et al., 2016; Novikova et al., 2017).

1.1 Contributions

To summarize, our completed and proposed contributions of this thesis are as follows:

1. Two novel feature-based models of sentence salience and an empirical evaluation on a stream summarization task.
2. Several novel deep learning architectures for word and sentence salience, a thorough evaluation of the linguistic and structural features critical to learning in the SDS task across a variety of genres, and adaptation to a news MDS task.
3. A novel training regime for generative models of text, called faithful generation, to ensure that the generated text does not misrepresent the conditioning input. We develop faithful generation models for both the data-to-text and text-to-text (i.e. summarization) settings.
4. A novel method of combining salience estimation based extractive summarization with abstractive generation in the faithful generation paradigm.

Together, these contributions provide a wide array of experiments and methodology for identifying summary worthy information and generating text that respects truth statements about that information. The hope is that these methods will lead to more reliable summaries without sacrificing the expressiveness of the generation algorithm. In the next section (section 2) we describe completed work on feature-based approaches to sentence salience estimation. In section 3, we describe completed and ongoing work on deep neural network models of salience estimation. We describe our planned approaches to faithful generation in section 4. Section 5 describes our research plan, before we finally conclude.

2 Feature-Based Models of Sentence Salience

Learning models of sentence salience in summarization is often difficult because summarization datasets are typically small (a few hundred training examples in many cases) and learning lexical feature weights directly is likely to run into issues of train-test distribution mismatch and overfitting. To get around this problem, we rely on heavily lexicalized components trainable on unlabeled data to produce scores that serve as unlexicalized features in our salience models. The canonical example of this is to use a language model trained on in-domain but non-summarization data to obtain average token perplexity scores for a sentence; the salience model only sees the average perplexity feature. In this section we describe our features in the context of a stream summarization task, but the features are themselves fairly generic and likely applicable in other task settings.

A second but important issue is that salience judgments do not occur in isolation. As we add content to the summary, the salience of our remaining inputs is likely to change based on redundancy and other factors. Unfortunately, adding summary/sentence interaction features introduces an element of exploration to training the model for now various summary configuration and candidate sentence pairs must be considered.

Our two proposed feature-based summarization models deal with this issue in slightly different ways. The first model, Salience-biased Affinity Propagation (SAP) (Kedzie et al., 2015), combines independent sentence level salience predictions into a clustering algorithm, Affinity Propagation (AP) (Frey and Dueck, 2007), that also considers sentence similarity to jointly select salient and non-redundant sentences for inclusion in the summary. In this model, for a sentence to be selected it must have a high salience estimate and also be representative of other sentences in the input.

Our second model, Learning-to-Summarize (L2S) (Kedzie et al., 2016), allows us to freely incorporate summary/sentence interaction features, as we train the salience model using the learning-to-search regime (Daumé et al., 2009; Chang et al., 2015) where learning takes place using different exploration policies. Using this method we can learn a salience model that makes good individual sentence selection choices (i.e. good local decisions) that also correlate to a good final summary (i.e. good global decisions).

In the next subsections, we will describe the stream summarization task and related work in detail, before moving on to the sentence features, and finally the models and their evaluations.

2.1 Stream Summarization Task and Related Work

In 2007, the Document Understanding Conference (DUC) piloted an "update" summarization task (Dang and Owczarzak) where summarization models were presented with an initial multi-document collection to summarize (similar to earlier DUC document sets for MDS), and then presented with another document set of related documents from a later time period. The evaluation criterion for the summary of the second document set was modified to not only be responsive to the content of the document set but also to focus on novel information that was not present in the initial document collection, hence this second summary was referred to as the update summary. Subsequent workshops at the Text Analysis Conference (TAC) allowed researchers to explore the effect of background information and redundancy measures on the update summarization task (typically in an unsupervised manner) (Chen et al., 2008; He et al., 2008; Mohammad et al., 2008; Zhang et al., 2008) as well as research into automatic evaluation measures to account for novelty (Conroy et al., 2011).

Stream summarization can be understood as a generalization of the update summarization

"hurricane sandy"	**"boston marathon bombing"**
[10/23 8:20pm] Sandy strengthened from a tropical depression into a tropical storm	[04/15 7:31pm] Authorities are investigating a report of two explosions at the finish line of the race.
[10/23 8:20pm] 2 pm Oct 23 Sandy moving north-northeast at 4 knots	[04/15 8:29pm] Some Boston Transit system service has been halted.
[10/23 8:53pm] forecast track uncertain	[04/16 12:37am] Police confirmed another explosion at the JFK Library.
[10/25 12:20am] In Jamaica damage was extensive	

Table 1: Example event queries in bold, and several example nuggets below. Nugget timestamps are shown in brackets.

task, where the number of updates is determined by the summarization model. In the most general sense, the stream summarization task requires a summarization model to process a stream of text data and produce text updates that describe the most important information in the stream as information passes through it. For our evaluations, we further ground this task in a crisis-monitoring scenario, as was used in the Temporal Summarization (TS) track of the Text Retrieval Conference (TREC) (Aslam et al., 2015, 2016).

In the disaster summarization scenario proposed by TREC, participant models received a brief query string q describing a natural or man-made disaster, and the model was expected to process a time-ordered stream of documents relevant to the query, extracting sentences that were likely to contain important facts about the event. Each query corresponded to a real-life disaster that was significant enough to have an associated entry in Wikipedia.

For each query, human annotators also collected a reference set of important facts, which we refer to as *nuggets*, from the revision history of that query's associated Wikipedia page. Nuggets consist of a piece of reference text and timestamp for when this piece of information first appeared in the Wikipedia revision history. See Table 1 for example queries and nuggets.

If a sentence s expresses the same piece of information as a nugget text n we say that s contains n or $n \in s$. Models are rewarded when they extract sentences that contain novel nuggets. Models are penalized for selecting sentences that are irrelevant (i.e. contain no nuggets) or contain nuggets already covered by previously extracted sentences. The official TS track relied on two primary metrics to evaluate a predicted extract summary: the expected gain, or $\mathbb{E}[\text{GAIN}]$, and COMPREHENSIVENESS (COMP.). $\mathbb{E}[\text{GAIN}]$ is the average number of novel nuggets contained in each sentence extracted by the model and can roughly be thought of as the nugget precision. COMP. is simply the recall of a given query's nuggets. Latency penalized metrics are also computed where the importance of a nugget decays over time. E.g. if a system recovers the nugget "25 people were reported injured," several days after this fact was first reported, it will receive less credit for it than the system that emits that nugget an hour after it enters the document stream. See Aslam et al. (2014) for more details on this decay factor. Intuitively, latency penalized metrics capture the idea that stale information in a rapidly evolving disaster is less useful and possibly distracting.

Previous approaches to this task have considered using hand tuned query relevance and novelty thresholds (Xu et al., 2013), linear models of salience and novelty with tuned thresholds for sentence selection (Guo et al., 2013), and learning adaptive rank cutoffs (McCreadie et al., 2014, 2015), and in general these approaches use cascades of threshold rules to extract sentences. By comparison, our proposed SAP model uses biased clustering to jointly consider salience and redundancy when making sentence selection decisions. The L2S model directly integrates redundancy as a feature in a salience model, in effect, answering the salience and similarity question simultaneously. SAP was the top performer at the 2014 TS track and fifth place in 2015; L2S was the fourth-place performer in 2015.

2.2 Features for Sentence Salience Estimation

The streaming summarization problem is difficult precisely because the context is constantly shifting. We cannot rely solely on word frequency because the counts of particular n-grams will be shifting throughout the document stream. Instead we compute several groups of sentence features that are specifically helpful for the streaming nature of the task. Because our models were developed over several years of the TS track, some features were not used or available to certain models. If a feature is only used in one model, we list that model's name in parenthesis next to the feature group. For more details see Kedzie et al. (2015) and Kedzie et al. (2016).

- **Surface Features.** These features include sentence length, the average number of capitalized words, document position, sentence length in words, and the average number of named entities.
- **Query Features.** We employ query match features that count frequency of query term matches in the sentence. We also do a simple query expansion using WordNet (Miller, 1995) to find synonyms, hypernyms, and hyponyms for the query *event type* and compute a similar term match count with the expansion.
- **Language Model Score.** We compute the average token log likelihood under two 5-gram Kneser-Ney language models (Kneser and Ney, 1995), one trained on newswire documents and one trained on a query *event type* specific sub-corpus of Wikipedia.
- **Geographic Relevance (SAP).** We compute several estimates of the distance of the events described in a sentence to the real world location of the event that corresponds to the query.
- **Temporal Relevance (SAP).** We compute rolling average TF-IDF scores for the last 24 hours to capture spikes in event activity.
- **Document Frequency (L2S).** We compute the hour-to-hour change in document frequency of the document stream to measure periods of heightened event activity.
- **Stream Language Model Score (L2S).** We compute the average unigram probability of several classes of words (non-stopwords, persons, places, and locations) whose word counts are updated with each new document in the stream.
- **Update Similarity (L2S).** We compute several measures of similarity between the previously extracted sentences and a candidate sentence.
- **Nugget Probability (L2S).** We compute the probability that a sentence contains a nugget using a classifier trained on human judgements using n-gram features.
- **Single Document Summarization Rankings (L2S).** We compute sentence rankings under several unsupervised extractive single document summarization algorithms.

2.3 Model 1: Salience-Biased Affinity Propagation Clustering

Identifying potential updates from the document stream is hard in part because we may not have enough context to use word frequencies as a reliable proxy for salience. Our first proposed method accounts for this by processing the stream in hourly batches, i.e. we collect all the sentences from the last hour and then decide which sentences if any to add to the update summary. The trade-off we make is that fast breaking events may not immediately be covered by the summarizer. The benefit is that we can now decompose our salience estimation into two components, a regression estimate of salience for each sentence individually, and a set of pairwise factors that describe how representative a sentence is of the other batch items.

We use an exemplar-based clustering algorithm called Affinity Propagation (AP) (Frey and Dueck, 2007) to systematically combine these unary and pairwise factors into a joint selection of cluster centers, called exemplars, that we consider as candidate updates. A sentence that has high overall similarity to the other sentences in the batch is likely to be an exemplar and more so if its salience estimation is high relative to the batch.

Finally, given a set of exemplar sentences, we remove any sentences whose similarity to any previous update is above a threshold. The remaining exemplars are added to the update summary. See Kedzie et al. (2015) for the full algorithm.

2.3.1 Salience Estimation

When we were developing this salience model, we did not have access to many human judgements of query or nugget relevance, and so we relied on an automatic measure of sentence salience computable directly from the nugget and sentence texts. Given a query q, sentence text s, and a query's nugget texts $n \in \mathcal{N}(q)$, we define the sentence salience y as

$$y = \max_{n \in \mathcal{N}(q)} \textsc{Similarity}(s, n)$$

where $\textsc{Similarity}(\cdot, \cdot)$ is the cosine similarity of a low-dimensional representation of the sentence and nugget text. We used the weighted matrix factorization method of Guo and Diab (2012) which projects a text's sparse high-dimensional bag-of-words representation into a dense, low-dimensional vector.

We use a Gaussian process regression model (Rasmussen, 2004) to predict y from s *without* knowledge of the nuggets. For each feature group g in subsection 2.2 we create a separate radial basis function (RBF) kernel and the actual Gaussian process is specified by the summation kernel $k(s, s') = \sum_g k_g(s, s')$ of the individual group kernels.

2.3.2 Salience-biased Affinity Propagation

Affinity Propagation (AP) is a factor-graph based clustering method that simultaneously selects the most representative data points to be cluster centers, referred to as *exemplars*, and maps the remaining data points to exactly one of those exemplars (Frey and Dueck, 2007). The exemplar mappings determine the clusters. AP has a number of nice properties for extractive summarization. First, as an exemplar based clustering method, the cluster centers are guaranteed to be an actual sentence observed in the input, and not an abstract mathematical object like a mean in the input feature space. Additionally, the number of clusters that result is adaptive and based on the energy of the factor graph configurations; as we have to summarize many hours of stream data of varying density, we can avoid having to heuristically propose a number of cluster centers that works across a broad range of stream conditions.

	Rouge-1			Rouge-2		
System	Recall	Precision	F_1	Recall	Precision	F_1
Sap	**0.282**	**0.344**	**0.306**	**0.045**	**0.056**	**0.049**
Ap	0.245	0.285	0.263	0.033	0.038	0.035
Rs	0.230	0.271	0.247	0.031	0.037	0.034
Hac	0.169	0.230	0.186	0.017	0.024	0.019

Table 2: System ROUGE performance.

Typically, all inputs to AP clustering are equally likely *a priori* to be selected as exemplars. In our case, we have some prior beliefs about the importance of a given datapoint as expressed by our salience predictions $\hat{y}_i$. By replacing the self-affinity potentials with our salience predictions and the pairwise potentials with our semantic similarity function from the previous section we obtain the salience-biased affinity propagation model. When we estimate a sentence to be more salient it is more likely *a priori* to form a cluster center. When that sentence is also highly similar to other sentences in the batch, it collects support from those sentences, further increasing its likelihood of being assigned as an exemplar.

2.3.3 Data

The document streams come from the news portion of the 2014 TREC KBA Stream Corpus (Frank et al., 2012), which contains hourly crawls of the web covering a roughly two year span from 2011 to 2013. Event queries and their nuggets were taken from the data prepared for the 2013 and 2014 TREC TS tracks. This data contained 25 events and their query strings, time period of interest, and event type. Additionally, each event was associated with anywhere from 50 to several hundred timestamped nugget texts. For details on the creation of this dataset see Aslam et al. (2014, 2015). From the larger KBA Stream Corpus we created event specific document streams by filtering out any documents that did occur in the period of interest and contain all the query words of the corresponding event.

2.3.4 Experiments

Of the 25 events in the TREC TS data, 24 are covered by the news portion of the TREC KBA Stream Corpus. From these 24, we set aside three events to use as a development set. All system salience and similarity threshold parameters are tuned on the development set to maximize Rouge-2 F_1 scores. We train a salience model for each event using 1000 sentences randomly sampled from the event's document stream. We perform a leave-one-out evaluation of each event. At test time, we predict a sentence's salience using the average predictions of the 23 other models.

Since we lacked gold judgments about what sentences contain which nuggets we perform an automatic evaluation using Rouge (Lin, 2004). We create reference summaries for each query by concatenating all of its nugget texts. Since there are no fixed summary lengths, and depending on the severity of the event, the reference summaries can vary greatly in length. To account for this, we report Rouge recall, precision and F_1 score. We also approximate the manual evaluation of the official TREC TS track by automatically mapping sentences to nuggets if their semantic similarity is a above a threshold. We report $\mathbb{E}\left[\text{Gain}\right]$, Comp., and their F_1 score across a sweep of similarity threshold values from zero to one, with values closer to one a more conservative estimate of performance.

We refer to our approach as SAP and compare to several baselines. The first is our full system but with uniform salience scores, i.e. the default AP clustering algorithm. We refer to this method as AP. The second is to rank all sentences in each batch in order of decreasing predicted salience, and sequentially add each sentence to the update summary, omitting any sentences with similarity to previous updates above a threshold. This method is referred to as RS for rank by salience. Finally, we compare against another clustering algorithm, hierarchical agglomerative clustering. In this method, sentences are first clustered, and then centers are determined by the sentence with highest cosine similarity to the cluster mean. Sentences are added to the update summary in time order, removing sentences that are highly similarity to previous updates in the same manor as the RS method. We refer to this method as HAC.

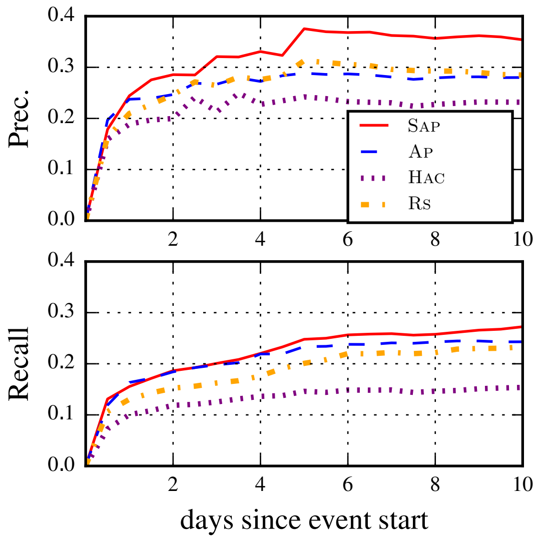

Figure 1: System ROUGE-1 performance over time.

2.3.5 Results

Table 2 shows our results for system output samples against the full summary of nuggets using ROUGE. This improvement is statistically significant for all n-gram precision, recall, and F_1 scores at the $\alpha = .01$ level using the Wilcoxon signed-rank test. SAP maintains its performance above the baselines over time as well. Figure 1 shows the ROUGE-1 scores over time. We show the difference in unigram precision (bigram precision is not shown but it follows a similar curve). Within the initial days of the event, SAP is able to take the lead over the over systems in ngram precision. The SAP model is better able to find salient updates earlier on; for the disaster domain, this is an especially important quality of the model. Moreover, the SAP recall is not diminished by the high precision and remains competitive with AP. Over time SAP's recall also begins to pull away, while the other models start to suffer from topic drift.

Figure 2 shows the $\mathbb{E}\,[\text{GAIN}]$ and COMP. across a range of similarity thresholds, where thresholds closer to 1 are more conservative estimates. The ranking of the systems remains constant across the sweep with SAP beating all baseline systems. Predicting salience in general is helpful for keeping a summary on topic as the RS approach out performs the clustering only approaches on $\mathbb{E}\,[\text{GAIN}]$. When looking at the COMP. of the summaries AP outperforms SAP. The compromise encoded in the SAP objective function, between being representative and being salient, is seen clearly here where the performance of the SAP methods is lower bounded by the salience focused RS system and upper bounded by the clustering only AP system. Overall, SAP achieves the best balance of these two metrics.

2.4 Model 2: Learning-to-Summarize

The SAP model has two primary shortcomings on the stream summarization task. First, by processing sentences in hourly batches, we potentially incur latency penalties if important information enters the stream at the start of the hour. Ideally we would process new documents as soon as possible in order to minimize the negative effects of latency. Second, the salience regression models were statically trained without exploration. If we had some sort of exploration during training, we could take advantage of features between previous sentence selection decisions and the current state of the update summary (something we didn't model in SAP).

In order to train with exploration, we need either many possible trajectories (i.e. document streams and associated sentence extraction labels) for all of the plausibly good summaries or some tractable oracle summarizer that we can use to complete a trajectory from any partially completed prefix of extraction decisions. Since we lack even one ground truth trajectory, we opt to use an oracle summarizer, which we refer to as the oracle policy, in keeping with the learning-to-search literature.

The oracle policy π^* has knowledge of what nuggets, if any, are present in each sentence in the document stream, and its behavior is quite simple: it processes the document stream sequentially and when it encounters a sentence with a novel nugget, it adds that sentence to the update summary.

As we stated before, training only on a single oracle run over document stream would be sub-optimal because the oracle is perfect and it would finish recovering all of the nuggets quite quickly and then do nothing for the remainder of the stream. In practice, our learned model is likely to make mistakes, missing the first few appearances of a nugget but hopefully we recover them as repetition in the stream makes them more likely to be selected. If we only followed the oracle's first best pass, it would not help us learn to recover from errors.

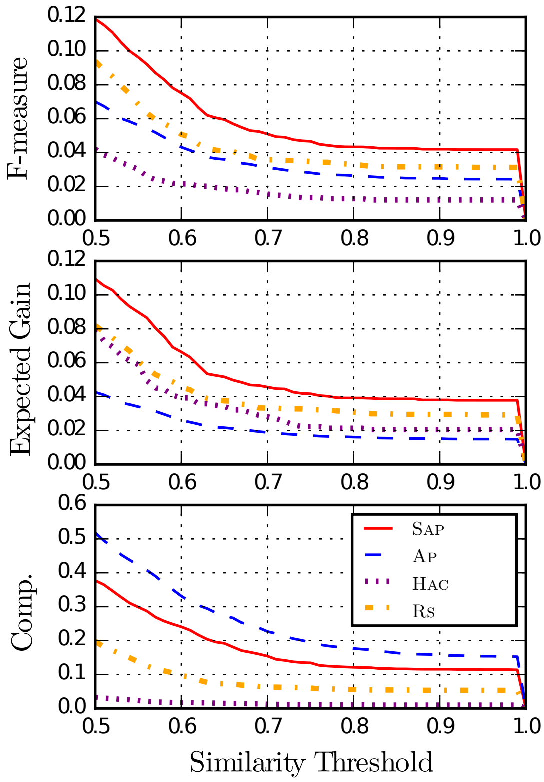

Figure 2: Expected Gain and Comprehensiveness performance.

To make better use of the oracle, we adopt the locally optimal learning to search (LOLS) regime (Chang et al., 2015), one of a family of learning-to-search algorithms. In order to formally describe the algorithm, we first introduce some notation. We define the stream summarization task as a Markov decision process. We observe a sequence of states $s_t \in \mathcal{S}$ which correspond to observing the first t sentences in the stream along with the first $t-1$ extraction decisions. A policy $\pi \in \prod$ maps states to actions where the action space $\mathcal{A} = \{a_0, a_1\}$ contains two actions with a_0 indicating "skip the current sentence" and a_1 indicating "extract the current sentence." Transitions between states are deterministic, i.e. the t-th sentence in the stream is appended to the update summary if a_1 was selected and the next stream sentence is presented to the summarizer. A trajectory $(s_1, a_1), \ldots, (s_n, a_n)$ of state-action tuples implicitly defines an extract label sequence $\mathbf{y} \in \{0,1\}^n$ for a stream of n sentences. Additionally, we assume possession of a non-negative loss function $\ell(\mathbf{y}, \mathbf{y}^*)$ which measures the discrepancy between the predicted extract sequence $\mathbf{y}$ and a reference sequence $\mathbf{y}^*$. The model policy interacts with states through a feature map $\phi : \mathcal{S} \rightarrow \mathbb{R}^d$ and a matrix of associated feature weights $\mathbf{W} \in \mathbb{R}^{2\times d}$ via $\pi(s_t) = \arg\min_{a\in\mathcal{A}} \mathbf{W}_a \cdot \phi(s_t)$.

In the LOLS regime, model training works by exploring the state space in two phases.

Consider a case where the document stream consists of n sentences. The first phase, called the *roll-in*, uses the current model policy $\hat{\pi}$ (which is initially random) to explore t steps into the stream, i.e. starting from state s_1, we repeatedly apply $\hat{\pi}$ and take its predicted action, stopping at state s_t. In the second phase we complete the trajectory using either the oracle π^* or the model $\hat{\pi}$ (the probability β of using π^* over $\hat{\pi}$ is a hyperparameter). Let π^{out} be the roll-out policy (either π^* or $\hat{\pi}$) , and let $\mathbf{y}^{(s_t,a)}$ be the extract label sequence implied by the completed trajectory of using $\hat{\pi}$ to navigate to s_t, taking action a, and then using π^{out} to complete the trajectory from s_{t+1} to the end state (i.e. the end of the document stream).

We also associate with $\mathbf{y}^{(s_t,a)}$ a non-negative cost $c(s_t, a)$ which connects the global discrepancy of the resulting structured loss to the local action decisions via the following formula

$$c(s_t, a) = \max_{a' \in \mathcal{A}} \ell(\mathbf{y}^{(s_t,a)}, \mathbf{y}^*) - \ell(\mathbf{y}^{(s_t,a')}, \mathbf{y}^*).$$

The cost of the locally optimal action will be zero, while the the locally sub-optimal action will have a cost equal to the difference between the structured losses associated with the sub-optimal and optimal actions as completed by π^{out}.

For each epoch of training, we select a training stream and for each sentence in the stream we perform a roll-in/roll-out, collecting the associated costs at each step into a dataset $\mathcal{D} = \left\{\left(\phi(s_1), c(s_1, a_0), a_0\right), \ldots, \left(\phi(s_T), c(s_T, a_1), a_1\right)\right\}$. After computing all costs for a given stream $\mathcal{S}(q)$, we update $\hat{\pi}$ to minimize the mean squared error of the sampled costs, i.e. we minimize the function

$$\mathcal{L}(\mathbf{W}) = \frac{1}{|\mathcal{D}|} \sum_{\phi(s_t), c(s_t,a), a \in \mathcal{D}} \Big(c(s_t, a) - \mathbf{W}_a \cdot \phi(s_t, a)\Big)^2$$

with respect to $\mathbf{W}$ using stochastic gradient descent.

See Kedzie et al. (2016) for the full training algorithm. The benefit of using $\hat{\pi}$ to obtain the roll-in trajectory is that the observed state spaces are more likely to be similar to ones visited on test data since compounding error in $\hat{\pi}$ will also effect the states visited at training time and thus minimizing train/test distribution mismatch. Randomly selecting π^* versus $\hat{\pi}$ is similarly helpful for ensuring that the completed trajectory is somewhat realistic (i.e. when using $\hat{\pi}$) but suitable for imitation due to the more optimal choices of the oracle. The theoretical motivations of LOLS can be found in greater depth in Chang et al. (2015).

2.4.1 Oracle Policy and Loss Function

We use a greedy oracle that selects sentences that contain novel nuggets. This oracle will achieve an optimal COMP. score, i.e. it will obtain every possible novel nugget in the roll-out phase. In order to implement a loss function ℓ we need a reference extract label sequence $\mathbf{y}^*$ for which to compare to the one obtained by the model policy, $\hat{\mathbf{y}}$. We set $\mathbf{y}^*_t = 1$ if the oracle policy would have selected that sentence given the prefix of predicted extractions $\hat{\mathbf{y}}_{1:t-1}$, i.e. if the t-th sentence contained a nugget not contained in any previous sentence extracted by $\hat{\pi}$, the reference extract would contain it.

We design our loss function to penalize policies that severely over- or under-generate. Given the reference and predicted extract summaries, we define the loss as the complement of the Dice coefficient between the decisions,

$$\ell(\mathbf{y}^*, \hat{\mathbf{y}}) = 1 - 2 \times \frac{\sum_i \mathbf{y}^*_i \hat{\mathbf{y}}_i}{\sum_i \mathbf{y}^*_i + \hat{\mathbf{y}}_i}.$$

This encourages not only local agreement between policies (the numerator of the second term) but that the learned and oracle policy should generate roughly the same number of updates (the denominator in the second term).

2.4.2 Salience Estimation

The model policy accesses a state-action tuple through the feature function ϕ, upon which we learn a linear function of the features to predict the anticipated cost of skipping or extracting the sentence. We can interpret the negative cost of selecting a sentence as its salience to the event query. Since a state s_t encodes the first t sentences and $t - 1$ extraction decisions made while processing the document stream, ϕ can represent a broader range of features than a salience model trained on static sentence salience judgements. In particular, we have several similarity features between the current sentence under consideration and the current state of the update summary. See subsection 2.2 for the complete list of features.

Many of our features are helpful for determining the importance of a sentence with respect to its document. However, they are more ambiguous for determining importance to the event as a whole. To compensate for this, we leverage two features which we believe to be good global indicators of update selection: the summary content probability and the document frequency. These two features are proxies for detecting (1) a good summary sentences (regardless of novelty with respect to other previous decisions) and (2) when an event is likely to be producing novel content. Therefore, we also compute feature conjunctions of all features in subsection 2.2 with the nugget probability and document frequency separately, as well as together.

2.4.3 Data

We used the official TREC 2015 Temporal Summarization dataset which was expanded from the previous year (on which we evaluated the SAP model). We use an expanded version of the stream corpus used in subsection 2.3 collected by Frank et al. (2012). The event query dataset was also expanded to 44 total queries with coverage in the stream stream corpus. Nuggets for the additional events were also collected. Additionally, pooled output from the 2015 TS performer submissions was manually labeled by NIST to obtain nugget relevance judgements. We used these judgements to build the nugget probability feature and to add noisy labels to the corpus.

Because of the large size of the corpus and the limited size of the manual annotation pool, many good candidate sentences were not manually reviewed. Less than 1% of the sentences in our relevant document streams received manual review. In order to increase the amount of data for training and evaluation of our system, we augmented the manual judgements with automatic or "soft" matches. A separate gradient boosting classifier was trained for each nugget with more than 10 manual sentence matches. Manually matched sentences were used as positive training data and an equal number of manually judged non-matching sentences were used as negative examples. Ngrams (1-5), percentage of nugget terms covered by the sentence, semantic similarity of the sentence to nugget were used as features, along with an interaction term between the semantic similarity and coverage. When augmenting the relevance judgments with these nugget match soft labels, we only include those that have a probability greater than 90% under the classifier. Overall these additional labels increase the number of matched sentences by 16 fold.

2.4.4 Experiments

To evaluate our model, we randomly select five events to use as a development set and then perform a leave-one-out style evaluation on the remaining 39 events. Even after filtering, each training query's document stream is still too large to be used directly in our combinatorial search space. In order to make training time reasonable yet representative, we downsample

	Unpenalized			Latency-Penalized			Num.
	$\mathbb{E}$[GAIN]	COMP.	F_1	$\mathbb{E}$[GAIN]	COMP.	F_1	Updates
SAP	$\mathbf{0.119}^{c}$	0.09	0.094	0.105	0.088	0.088	8.3
COS	0.075	0.176^{s}	0.099	0.095	0.236^{s}	0.128^{s}	$145.6^{s,f}$
L2S	0.097	$\mathbf{0.207}^{s,f}$	0.112	0.136^{c}	$\mathbf{0.306}^{s,c,f}$	0.162^{s}	$89.9^{s,f}$
L2SCOS	$0.115^{c,l}$	0.189^{s}	$\mathbf{0.127}^{s,c,l}$	$\mathbf{0.162}^{s,c,l}$	0.276^{s}	$\mathbf{0.184}^{s,c,l}$	$29.2^{s,c}$

Figure 3: Average system performance and average number of updates per event. Superscripts indicate significant improvements ($p < 0.05$) between the run and competing algorithms using the paired randomization test with the Bonferroni correction for multiple comparisons (s: SAP, c: COS, l: L2S, f: L2SCOS). F_1 is the harmonic mean of $\mathbb{E}$[GAIN] and COMPREHENSIVENESS.

each stream to a length of 100 sentences. The downsampling is done uniformly over the entire stream. This is repeated 10 times for each training event to create a total of 380 training streams. In the event that a downsample contains no nuggets (either human or automatically labeled) we resample until at least one exists in the sample.

In order to avoid overfitting, we select the model iteration for each training fold based on its performance (in harmonic mean of $\mathbb{E}$ [GAIN] and COMP.) on the development set.

We refer to our "learning to summarize" model in the results as L2S. We compare our proposed model against several baselines and extensions.

One of the top performing systems at TREC 2015 was a heuristic method that only examined article first sentences, selecting those that were below a cosine similarity threshold to any of the previously selected updates (Raza et al., 2015). We implemented a variant of that approach using the latent-vector representation used throughout this work (Guo and Diab, 2012). The development set was used to set the threshold. We refer to this model as COS.

We also compare this model to a simplified variant of the SAP model where we use the summary nugget probability feature as the salience estimator (the Gaussian process models were difficult to scale to the expanded dataset). The time window size, similarity threshold, and an offset for the cluster preference are tuned on the development set. As in the COS model, a similarity threshold is used to filter out updates that are too similar to previous updates (i.e. previous clustering outputs). We refer to this method as SAP.

Our final baseline, which we refer to as L2SCOS, we run L2S as before, but filter the resulting updates using the same cosine similarity threshold method as in COS. The threshold was also tuned on the development set.

2.4.5 Results

Results for system runs are shown in Figure 3. On average, L2S and L2SCOS achieve higher F_1 scores than the baseline systems in both latency penalized and unpenalized evaluations. For L2SCOS, the difference in mean F_1 score was significant compared to all other systems (for both latency settings).

SAP achieved the overall highest $\mathbb{E}$ [GAIN], partially because it was the tersest system we evaluated. However, only COS was statistically significantly worse than it on this measure.

In comprehensiveness, L2S recalls on average a fifth of the nuggets for each event. This is even more impressive when compared to the average number of updates produced by each system (Figure 3); while COS achieves similar comprehensiveness, it takes on average about 62% more updates than L2S and almost 400% more updates than L2SCOS. The output size of

	latency-penalized		
	$\mathbb{E}$[GAIN]	COMP.	F_1
COS	0.095	**0.236**	0.128
L2SFS	0.164	0.220	0.157
L2SCOSFS	**0.207**	0.18	**0.163**

Figure 4: Average system performance. L2S-FS and L2S-COS-FS runs are trained and evaluated on first sentences only (like the COS system). Unpenalized results are omitted for space but the rankings are consistent. F_1 is the harmonic mean of $\mathbb{E}$[GAIN] and COMPREHENSIVENESS.

COS stretches the limit of the term "summary," which is typically shorter than 145 sentences in length. This is especially important if the intended application is negatively affected by verbosity (e.g. crisis monitoring).

Since COS only considers the first sentence of each document, it may miss relevant sentences below the article's lead. In order to confirm the importance of modeling the oracle, we also trained and evaluated the L2S based approaches on first sentence only streams. Figure 4 shows the latency penalized results of the first sentence only runs. The L2S approaches still dominate COS and receive larger positive effects from the latency penalty despite also being restricted to the first sentence. Clearly having a salience model (beyond similarity) of what to select is helpful. Ultimately we do much better when we can look at the whole document.

3 Deep Learning Models of Content Salience

Increasingly, deep neural network models are being used to perform sentence extractive summarization tasks. While these models are very flexible and allow for easy hierarchical representation learning, it is unclear what signals they are extracting from the data to make predictions. In this section, we describe our completed experiments teasing out the importance of different neural network designs for sentence level salience estimation (Kedzie et al., 2018). In particular, we experiment with several methods for encoding a sequence of word embeddings into a sentence embedding, and then in turn, mapping a sequence of sentence embeddings to sentence salience predictions. We introduce several simplifications to existing models in the literature, and show their effectiveness on an SDS task across news, personal narratives, workplace meetings, and medical journal article genres.

We also perform several diagnostic experiments and find impediments to learning robust models of sentence salience. In particular, the sentence position implicitly encoded in the models dominates the learning signal. While sentence position is certainly an important feature in news, not all domains or tasks will share this feature; we would also like to be able to design models that make their salience decisions primarily on lexical or topical content.

We believe that the sentence embedding representation is too coarse to make significant use of lexical information in the presence of less noisy position features, even when position is only implicitly represented by the model. To that end, we propose a new deep learning based SDS model that directly estimates individual word level salience scores, and a simple sentence selection and margin loss framework for learning. In this model, we augment the word embeddings (which only capture shallow lexical semantics) with embeddings representing other word features. Our initial experiments suggest that document frequency, and information theoretic accounts of surprisal (e.g. topic signatures) are also useful for the summarization task. We expect to show less dependence on the position features using the same ablation diagnostics we applied to our sentence level salience models. While previous work has estimated word importance using these features in a linear model (Hong and Nenkova, 2014), they were only able to take limited advantage of context features, i.e. taking a weighted average of word features to the left and right. By contrast, in our proposed model we can combine rich document specific contextual features, e.g. ELMO embeddings (Peters et al., 2018), in conjunction with these word features.

Additionally, we plan to adapt this word level salience model to a news MDS task; if it is less dependent on sentence position, it should be more amenable to MDS where lexical centrality to the document cluster is possibly the dominant learning signal. The MDS version of the model will use an importance score aggregation step where word level scores accrue additional importance across documents using an attention mechanism.

In the next subsections we first briefly cover related work on sentence and word level salience estimation before covering the completed sentence level and proposed word level salience estimation experiments.

3.1 Related Work

Estimating sentence salience for summarization has often been approached as a sentence classification problem. Naïve Bayes (Kupiec et al., 1995; Teufel, 1997; Osborne, 2002), maximum entropy (Osborne, 2002), and support vector machine (Hirao et al., 2002) classifiers have all been applied to predict whether a sentence should be included in an extract summary. Sentence position features were a strongly predictive signal in all of these works. Lexical features were more varied in their use; e.g., Kupiec et al. (1995) checked for the presence of key phrases from a manually curated list, Osborne (2002) experimented with automatically extracted bigram features, while Hirao et al. (2002) used average TF-IDF weights to capture important lexical content.

While the previous works all estimated sentence salience independently, a variety of structured prediction methods have also been explored to jointly estimate the salience of sentences in a document or document cluster. These include hidden Markov models (HMMs) (Conroy and O'leary, 2001), conditional random fields (CRFs) (Shen et al., 2007), large margin classifiers (Martins and Smith, 2009), and structured support vector machines (Berg-Kirkpatrick et al., 2011; Sipos et al., 2012; Durrett et al., 2016). While positional or discourse features are important to all of these methods, Martins and Smith (2009), Berg-Kirkpatrick et al. (2011), and Durrett et al. (2016) also learn lexical feature weights as part of their larger model. Graph random walk based ranking has been another prominent method for estimating salience of a collection of sentences jointly (Erkan and Radev, 2004; Mihalcea and Tarau, 2004); typically the sentence graphs are constructed using the cosine similarity between a sentence's TF-IDF weighted bag-of-words although other methods of weighting graph edges have been used. Additionally, independent sentence level priors about salience can also be incorporated by adjusting the probability of restarting the random walk (Erkan, 2006; Liu et al., 2008).

Deep learning methods have become the *de facto* standard approach to many NLP problems, especially when there exists plentiful labeled data. There has been a flurry of recent work on sentence extractive single document summarization of news using a variety of neural network architectures (Cheng and Lapata, 2016; Nallapati et al., 2016b,a; Narayan et al., 2018). These models have hierarchical representations of the document, using either recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to encode word embeddings into sentence embeddings which are then fed into an RNN based sentence extractor. Unlike prior structured approaches, finding the optimal label sequence in these models is intractable. However, the richer word and sentence representations often yield better performance even with a simpler inference method.

There is also prior work on estimating word importance directly (as opposed to learning word weights as a means to estimate sentence salience). For example, Yih et al. (2007) learn to predict the likelihood of a term appearing in a summary using a maximum entropy classifier with several document frequency and position features. Hong and Nenkova (2014) extend these features to consider word type (e.g. named-entity type), background information like the probability of the word occurring in a large collection of New York Times abstracts, and whether or not the word occurs in an automatic summary (using an unsupervised summary method). There does not appear to be much literature on extending this work with deep learning, a gap we hope to fill with our proposed word level model. Outside of summarization, Sheikh et al. (2016) found that learning importance scores for words in a weighted bag-of-words model outperformed TF-IDF based approaches to text classification tasks like sentiment and topic detection.

3.2 Deep Learning Models of Sentence Salience

Given the diversity of neural architectural choices, a best practices for sentence extractive summarization has yet to emerge. In this section we ask what architecture design choices matter for single document summarization across a variety of domains.

We begin by defining some terminology. A sentence is represented as an sequence of words $s = w_1, w_2, \ldots, w_{|s|}$, where each word is drawn from a finite vocabulary $\mathcal{V}$ and $|s|$ is the length of sentence s in words. Similarly, a document $d = s_1, s_2, \ldots, s_{|d|}$ is a sequence of sentences, where $|d|$ is the size of the document in sentences.

We treat sentence extractive summarization as a sequence tagging problem: given a document d, we want to assign an associated binary tag sequence $\mathbf{y} \in \{0, 1\}^{|d|}$ such that the corresponding set of extracts $\mathcal{E} = \{s_i \in d \mid \mathbf{y}_i = 1\}$ is a suitable summary of the document. Typically, it is assumed that the size of the extract summary in sentences is much smaller than the input document, i.e. $|\mathcal{E}| \ll |d|$. It is also common to enforce a word budget c such that $\sum_{s\in\mathcal{E}} |s| \leq c$.

A typical deep learning model will build up a hierarchical representation of each sentence, starting at the word level, and then composing an arbitrarily long sequence of word representations into a fixed length sentence representation. First the individual words are projected to fixed length vectors, or word embeddings. The sentence encoder network is then responsible for mapping word embedding sequences to fixed length sentence embeddings. Finally, the sentence extractor network produces a label sequence $\mathbf{y}$.

We explore several choices of encoder and extractor architecture from the literature (Cheng and Lapata, 2016; Nallapati et al., 2016a) as well as propose our own designs (Kedzie et al., 2018). For complete details of the different model architectures please see our paper. In the next sections, we describe the different encoder and extractor architectures, before discussing data, experiments, and evaluation.

3.2.1 Sentence Encoders

We explore three different sentence encoders commonly used in the literature.

1. **Averaging Encoder** A sentence embedding is obtained by averaging its associated word embeddings.

2. **RNN Encoder** We run a bidirectional RNN over the sentence's word embeddings, taking a concatenation of the final forward and backward output states as the sentence embedding. We use a gated recurrent unit (Cho et al., 2014) as the actual RNN cell.

3. **CNN Encoder** We use a CNN over a sentence's word embedding to extract n-gram features. Our implementation is similar to Kim (2014), i.e. one dimensional convolutions of varying width n-gram feature detectors with max pooling.

3.2.2 Sentence Extractors

A sentence extractor takes the encoder output, i.e. a sequence of sentence embeddings $\text{ENCODER}(d) = \text{ENCODER}(s_1), \ldots, \text{ENCODER}(s_{|d|}) = \mathbf{h}_1, \ldots, \mathbf{h}_{|d|}$, and produces an extract label sequence $\mathbf{y}$. The sentence extractor is essentially a discriminative classifier $p(\mathbf{y}_{1:|d|}|\mathbf{h}_{1:|d|})$. Prior neural network approaches (Nallapati et al., 2016a; Cheng and Lapata, 2016) to sentence extraction have assumed an auto-regressive model, leading to a semi-Markovian factorization of the extractor probabilities $p(\mathbf{y}_{1:|d|}|\mathbf{h}_{1:|d|}) = \prod_{i=1}^{|d|} p(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{h}_{1:|d|})$, where each prediction $\mathbf{y}_i$ is dependent on *all* previous $\mathbf{y}_j$ for all $j < i$. We compare two such models proposed by Cheng and Lapata (2016) and Nallapati et al. (2016a). A simpler approach that does not allow interaction among the $\mathbf{y}_{1:n}$ is to model $p(\mathbf{y}_{1:|d|}|\mathbf{h}_{1:|d|}) = \prod_{i=1}^{|d|} p(\mathbf{y}_i|\mathbf{h}_{1:|d|})$, which we explore in two proposed extractor models. Overall, we experiment with four different sentence extractor architectures which we now describe briefly.

1. **SummaRunner Extractor (Nallapati et al., 2016a)** The SummaRunner extractor is built around an RNN over sentence embeddings along with separate components to estimate contributions of document similarity, iterative summary similarity, and coarse and fine grained sentence position. Any one extraction decision $\mathbf{y}_i$ is dependent on all previously label decisions $\mathbf{y}_{<i}$.

2. **RNN Extractor (ours)** We propose a stripped down version of SummaRunner using only an RNN over sentence embeddings without any of the representations of document, summary, or position. The RNN output is fed to a sigmoid layer to predict the probability of extraction. All sentence extraction decisions are independent once the hidden layers of the RNN have been computed.

3. **Cheng & Lapata Extractor (Cheng and Lapata, 2016)** This extractor is built around a sequence-to-sequence model with distinct encoder and decoder RNNs over sentence embeddings (the encoder here is distinct from the sentence encoder over words). In effect the model processes the document twice. Predictions are made using the previous timestep's decoder output, and the current timestep's encoder output. Decoder inputs are weighted by the previous extraction probability, inducing a similar dependence structure to the SummaRunner model where any extraction decision $\mathbf{y}_i$ is dependent on all previous decsions $\mathbf{y}_{<i}$.

4. **Seq2Seq Extractor (ours)** Our simplified version of the Cheng & Lapata extractor is a standard sequence-to-sequence model (Bahdanau et al., 2014) with dot-product style attention (Luong et al., 2015). A concatenation of the decoder output and attention context are fed into a sigmoid layer to make predictions. Like the RNN extractors, sentence extraction decisions are independent once the RNN and attention layers have been computed.

Dataset	Train	Valid	Test	Refs
CNN/DailyMail	287,113	13,368	11,490	1
NYT	44,382	5,523	6,495	1.93
DUC	516	91	657	2
Reddit	404	24	48	2
AMI	98	19	20	1
PubMed	21,250	1,250	2,500	1

Table 3: Sizes of the training, validation, test splits for each dataset and the average number of test set human reference abstracts per document.

3.2.3 Data

We perform our experiments across six corpora from varying domains to understand how different biases within each domain can affect content selection. The corpora come from the news domain (CNN-DailyMail, New York Times, DUC), personal narratives domain (Reddit), workplace meetings (AMI), and medical journal articles (PubMed). See Table 3 for dataset statistics and Kedzie et al. (2018) for a complete description of the preprocessing. The news datasets are fairly common in the summarization literature and allow us to consider performance across large, medium, and small dataset sizes. The PubMed dataset contains similarly formal and position dependent text as newswire, and it allows us to examine the effect of these features in another genre. The Reddit and AMI corpus are less dependent on sentence position than the other datasets, however their smaller size makes them less useful as datasets for deep learning models. The AMI corpus is also transcribed speech which makes it a fairly unique summarization corpus.

Ground Truth Extract Summaries Typically, the reference summaries in our datasets are human written abstracts and therefore we cannot use them directly to learn a sentence extractive summarization model. Instead, we use them to construct gold label sequences by greedily optimizing ROUGE-1 with respect to the reference abstracts as in Nallapati et al. (2016a). We choose to optimize for ROUGE-1 rather than ROUGE-2 similarly to other optimization based approaches to summarization (Sipos et al., 2012; Durrett et al., 2016; Nallapati et al., 2016a) which found this to be the easier target to learn.

3.2.4 Experiments

We are interested in two questions. The first, more pragmatic question, is what are the best configuration of encoder/extractor architectures? We answer this by evaluating ROUGE recall performance across our six collected datasets. We also used METEOR (Denkowski and Lavie, 2014) but omit these numbers here for space since they show the same trends; see Kedzie et al. (2018) for the complete results. We perform the standard stochastic gradient descent based optimization (using the Adam update (Kingma and Ba, 2014)) of the weighted negative log likelihood of the gold extract labels to fit model parameters. We upweight the loss for positive labels (i.e. extractions) since they are relatively sparse compared to the negative labels.

The second question, is more diagnostic in nature: what signals in the data are driving model learning? We perform several experiments to find answers. We hypothesize that the lexical semantics encoded at the word embedding level will be important to subsequent sentence representations, and perform a comparison on learning with and without fine tuning of the embeddings. In both cases, embeddings are initialized with GLOVE embeddings pretrained on Wikipedia and Gigaword (Pennington et al., 2014).

We also hypothesize that certain classes of words will be more important to identifying salient content than others. We perform word ablation experiments where we alternately remove nouns, verbs, adjectives & adverbs, and function words from the sentence encoder input and compare performance to the non-ablated system. We expect that the nouns will be more important to content selection.

Our final experiments attempt to tease out the effect of structural features from the lexical. In this experiment, we shuffle the sentence order at training time. In this setup, we obfuscate features about which content was introduced in the article first, an important and well known bias in the news domain (Nenkova, 2005).

3.2.5 Results

Extractor	Encoder	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
Lead	–	24.4	32.3	21.5	**10.9**	2.0	9.3
RNN	Avg.	25.4	34.7	22.7	**11.4**	**5.5**	17.0
	RNN	25.4	34.9	22.6	**11.4**	**5.2**	16.6
	CNN	25.1	33.7	**22.7**	**12.8**	3.2	16.8
Seq2Seq	Avg.	**25.6**	**35.7**	**22.8**	**13.6**	**5.5**	**17.7**
	RNN	25.3	**35.9**	22.5	**12.0**	**5.3**	16.7
	CNN	25.1	35.1	**22.7**	**13.2**	2.9	16.9
Cheng & Lapata	Avg.	25.3	**35.6**	**23.1**	**13.6**	**6.1**	**17.7**
	RNN	25.0	**35.8**	**23.0**	**12.6**	**5.0**	16.7
	CNN	25.1	35.0	**23.0**	**13.4**	2.8	16.9
Summa Runner	Avg.	25.4	35.4	22.3	**13.4**	**5.6**	17.2
	RNN	25.2	35.5	22.1	**12.5**	**5.4**	16.5
	CNN	25.0	34.4	22.2	**12.3**	3.2	16.8
Oracle	–	36.2	48.9	31.8	16.2	3.9	25.0

Table 4: **Overall Results** ROUGE-2 recall results across all extractor/encoder pairs. Results that are statistically indistinguishable from the best system are shown in bold face.

The results of our main experiment comparing the different extractors/encoders are shown in Table 4 (Overall Results). Overall, we find no major advantage when using the CNN and RNN sentence encoders over the averaging encoder (see red Seq2Seq Avg. row). The best performing encoder/extractor pair either uses the averaging encoder (five out of six datasets) or the differences are not statistically significant.

When looking at extractors, the Seq2Seq extractor is either part of the best performing system (three out of six datasets) or is not statistically distinguishable from the best extractor (see again red Seq2Seq Avg. row).

Overall, on the news and medical journal domains, the differences are quite small with the differences between worst and best systems on the CNN/DM dataset spanning only .56 of a ROUGE point. While there is more performance variability in the Reddit and AMI data, there is less distinction among systems: no differences are significant on Reddit and every extractor has at least one encoder configuration that is indistinguishable from the best system on the AMI corpus. This is probably due to the small test size of these datasets.

Ext.	Emb.	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
RNN	Fixed	**25.4**	**34.7**	**22.7**	**11.4**	**5.5**	**17.0**
	Learned	25.2	34.3	**22.6**	**11.3**	**5.3**	16.4
Seq2Seq	Fixed	**25.6**	**35.7**	**22.8**	**13.6**	5.5	**17.7**
	Learned	25.3	**35.7**	**22.9**	**13.8**	**5.8**	16.9
C&L	Fixed	**25.3**	**35.6**	**23.1**	**13.6**	**6.1**	**17.7**
	Learned	24.9	35.4	**23.0**	**13.4**	**6.2**	16.4
Summa Runner	Fixed	**25.4**	**35.4**	**22.3**	**13.4**	**5.6**	**17.2**
	Learned	25.1	35.2	**22.2**	12.6	**5.8**	16.8

Table 5: **Word Embedding Learning** ROUGE-2 recall across sentence extractors when using fixed pretrained embeddings or when embeddings are updated during training. In both cases embeddings are initialized with pretrained GloVe embeddings using the averaging sentence encoder. When both rows are bolded, there is no signifcant performance difference.

Word Embedding Learning Given that learning a sentence encoder (averaging has no learned parameters) does not yield significant improvement, it is natural to consider whether learning word embeddings is also necessary. In Table 5 (Word Embedding Learning) we compare the performance of different extractors using the averaging encoder, when the word embeddings are held fixed or learned during training. In both cases, word embeddings are initialized with GLOVE embeddings trained on a combination of Gigaword and Wikipedia. When learning embeddings, words occurring fewer than three times in the training data are mapped to an unknown token (with learned embedding).

In all but one case, fixed embeddings are as good or better than the learned embeddings. This is a somewhat surprising finding on the CNN/DM data since it is reasonably large, and learning embeddings should give the models more flexibility to identify important word features.[1] This suggests that we cannot extract much generalizable learning signal from the content other than what is already present from initialization. Even on PubMed, where the language is quite different from the news/Wikipedia articles the GLOVE embeddings were trained on, learning leads to significantly worse results.

Ablation	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
all words	**25.4**	**34.7**	22.7	**11.4**	5.5	**17.0**
-nouns	25.3†	34.3†	22.3†	10.3†	3.8†	15.7†
-verbs	25.3†	34.4†	22.4†	10.8	5.8	16.6†
-adj/adv	25.3†	34.4†	22.5	9.5†	5.4	16.8†
-function	25.2†	34.5†	**22.9**†	10.3†	**6.3**†	16.6†

Table 6: **POS Tag Ablation** ROUGE-2 recall after removing nouns, verbs, adjectives/adverbs, and function words. Ablations are performed using the averaging sentence encoder and the RNN extractor. Bold indicates best performing system. † indicates significant difference with the non-ablated system.

[1] The AMI corpus is an exception here where learning *does* lead to small performance boosts, however, only in the Seq2Seq extractor is this difference significant; it is quite possible that this is an artifact of the very small test set size.

POS Tag Ablation It is also not well explored what word features are being used by the encoders. To understand which classes of words were most important we ran an ablation study, selectively removing nouns, verbs (including participles and auxiliaries), adjectives & adverbs, and function words (adpositions, determiners, conjunctions). The embeddings of removed words were replaced with a zero vector, preserving the order and position of the non-ablated words in the sentence. Ablations were performed on training, validation, and test partitions, using the RNN extractor with averaging encoder. Table 6 (POS Ablations) shows the results of the POS tag ablation experiments. While removing any word class from the representation generally hurts performance (with statistical significance), on the news domains, the absolute values of the differences are quite small (.18 on CNN/DM, .41 on NYT, .3 on DUC) suggesting that the model's predictions are not overly dependent on any particular word types. On the non-news datasets, the ablations have a larger effect (max differences are 1.89 on Reddit, 2.56 on AMI, and 1.3 on PubMed). Removing nouns leads to the largest drop on AMI and PubMed. Removing adjectives and adverbs leads to the largest drop on Reddit, suggesting the intensifiers and descriptive words are useful for identifying important content in personal narratives. Curiously, removing the function word POS class yields a significant improvement on DUC 2002 and AMI.

Extractor	Order	CNN/DM	NYT	DUC	Reddit	AMI	PubMed
Seq2Seq	In-Order	**25.6**	**35.7**	**22.8**	**13.6**	5.5	**17.7**
	Shuffled	21.7	25.6	21.2	**13.5**	**6.0**	14.9

Table 7: **Document Shuffling** ROUGE-2 recall using models trained on in-order and shuffled documents. Extractor uses the averaging sentence encoder. When both in-order and shuffled settings are bolded, there is no signifcant performance difference. Difference in scores shown in parenthesis.

Document Shuffling Sentence position is a well known and powerful feature for news summarization (Hong and Nenkova, 2014), owing to the intentional lead bias in the news article writing[2]; it also explains the difficulty in beating the lead baseline for single-document summarization (Nenkova, 2005; Brandow et al., 1999). In examining the generated summaries, we found most of the selected sentences in the news domain came from the lead paragraph of the document. This is despite the fact that there is a long tail of sentence extractions from later in the document in the ground truth extract summaries (31%, 28.3%, and 11.4% of DUC, CNN/DM, and NYT training extract labels come from the second half of the document). Because this lead bias is so strong, it is questionable whether the models are learning to identify important content or just find the start of the document. We conduct a sentence order experiment where each document's sentences are randomly shuffled during training. We then evaluate each model performance on the unshuffled test data, comparing to the model trained on unshuffled data; if the models trained on shuffled data drop in performance, then this indicates the lead bias is the relevant factor.

Table 7 (Document Shuffling) shows the results of the shuffling experiments. The news domains and PubMed suffer a significant drop in performance when the document order is shuffled. By comparison, there is no significant difference between the shuffled and in-order

[2] `https://en.wikipedia.org/wiki/Inverted_pyramid_(journalism)`

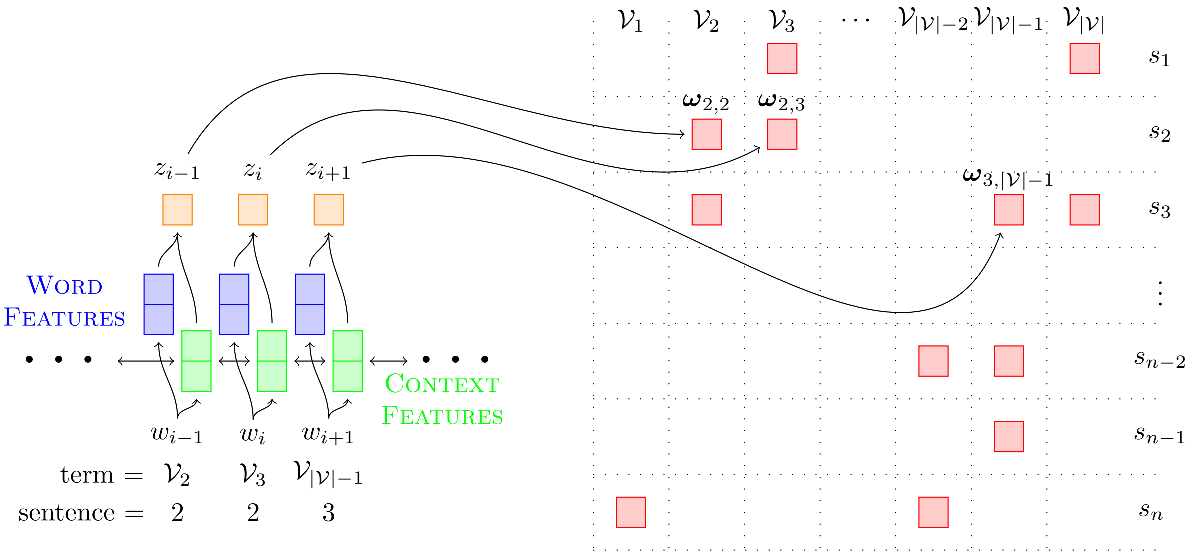

Figure 5: (Left) Word and context features are extracted from the flat token sequence representation to get word level importance scores z_i. (Right) Word level scores are aggregated into a sparse bag-of-words matrix with rows and columns corresponding to sentences and words respectively.

models on the Reddit domain, and shuffling actually improves performance on AMI. This suggest that position is being learned by the models in the news/journal article domain even when the model has no explicit position features, and that this feature is more important than either content or function words.

3.3 Word Importance Estimation in Deep Learning Models

Our previous experiments revealed that lexical semantics were not the main driver of learning in sentence extractive news summarization. One could plausibly argue that it is a feature, not a bug, that the structural signals in news are intentional and not to be avoided. However, we think more attention could be paid to estimating saliency at the word level.

Direct estimation of word importance has the following benefits. First, it provides increased explainability that the previously considered sentence extractive models do not have. Is a model predicting to extract because the sentence contains a discourse signaling phrase like "in conclusion," or is it the mention of an entity central to the story? The sentence extractor architectures can only show salience at the sentence level, making it hard to say anything specific about what words or phrases are contributing to this.

Moreover, the previous sentence level experiments revealed that different domains put salience on different word classes, e.g. adjectives and adverbs are particularly important for Reddit but less so for news. The availability of word level importance scores would help us to judge what models are likely to perform well on other domains based on which word classes they assign greater importance.

By focusing on word level estimation, we can also explore embeddings of features that have previously been shown to be useful for word importance estimation like syntax (Rose, 2005) or information theoretic notions of surprise and background information (Lin and Hovy, 2000; Hong and Nenkova, 2014; Louis, 2014). These features have yet do be explored in conjunction with deep neural network models for summarization. With RNNs we can also combine these independent word level features with representations of the surrounding document context.

Finally, word importance estimation can be used as a preprocessing step for abstractive summarization (Gehrmann et al., 2018). We would like to expand upon prior work in this area by also using a predicted word importance distribution as a form of supervised attention, i.e. the cumulative attention of an encoder-decoder abstractive summarization model should roughly match the distribution proposed by our word importance model.

3.3.1 Proposed Model

A document d is a sequence of m words $(w_1, w_2, \ldots, w_m) \in \mathcal{V}^m$ where $\mathcal{V}$ is a finite vocabulary. We define two mappings of words to dense vector representations. The first $\textsc{Features} : \mathcal{V} \to \mathbb{R}^{n_f}$ maps words to a concatenation of feature embeddings whose total dimension is of size n_f. The various components of the feature embeddings include the word's GLOVE embedding, as well as embeddings for sentence position, document frequency, topic signature, dependency role, and other word features that have been shown to be useful for summarization. The second mapping $\textsc{Context} : \mathcal{V} \to \mathbb{R}^{n_c}$ maps the word to its contextual embedding; here this corresponds to the output of ELMO (Peters et al., 2018) at that word's position in the document. The importance score z_i of a word w_i is the output of a feed-forward layer

$$z_i = \sigma\left(\mathbf{v}^T \begin{bmatrix} \textsc{Features}(w_i) \\ \textsc{Context}(w_i) \end{bmatrix} + b\right)$$

where $\mathbf{v} \in \mathbb{R}^{n_f + n_c}$ and $b \in \mathbb{R}$ are learned weight and bias parameters, and σ is the logistic sigmoid.

Next, we aggregate the flat token level scores z_i into a bag-of-words (BOW) representation for each sentence in the document. Let I_i be the set of indices of the flat word sequence corresponding to the words in i-th input sentence. Then, let $\boldsymbol{\omega}_i$ be the BOW representation of the i-th sentence with entries

$$\boldsymbol{\omega}_{i,j} = \begin{cases} 0 & \text{if } w_k \neq \mathcal{V}_j \text{ for all } k \in I_i \\ \sum_{k \in I_i} \mathbb{1}\{w_k = \mathcal{V}_j\} \cdot z_k & \text{otherwise} \end{cases}$$

for all $j \in \{1, \ldots, |\mathcal{V}|\}$. I.e. we create weighted BOW representations of the sentences using the word importance scores as the weights (summing the scores if a word occurs multiple times in the same sentence).

With the BOW representations in hand, we perform sentence selection using the algorithm presented in Figure 6 to obtain predicted extract indices $\hat{\mathbf{y}}$ and their associated overall summary score $\hat{\eta}$.

We can optimize this model using a margin loss, where given a gold extract sequence $\mathbf{y}$, we can compute the associated gold extract summary score η and then minimize the following loss function

$$\mathcal{L}_{ext}(\hat{\mathbf{y}}, \mathbf{y}; \boldsymbol{\theta}) = \max\left(0, 1 + \hat{\eta} - \eta\right)$$

with respect to the parameters $\boldsymbol{\theta}$ of the word importance predictor. In order to minimize the margin loss, the scores z_i must reflect the contributions of individual words to a sentence's rank for extraction, with words more predictive of summary inclusion receiving higher values.

Furthermore, can also introduce a supervised learning signal to the individual word importance scores by collecting labels ζ_i for each z_i such that $\zeta_i = 1$ if w_i occurs in any human

procedure BOWEXTRACTER($\boldsymbol{\omega}_{1:n}, \beta, \kappa$)
  $\boldsymbol{\omega}_i^{(1)} \leftarrow \boldsymbol{\omega}_i \quad \forall i \in \{1, \ldots, n\}$
  $\hat{\eta} \leftarrow 0$
  $t \leftarrow 0$
  while $\sum_{i=1}^{t} \kappa_{\hat{\mathbf{y}}_i} < \beta$ and $t < n$ **do**
    $t \leftarrow t + 1$
    $\hat{y}_t \leftarrow \operatorname{argmax}_{i \in \{1,\ldots,n\}} \sum_{j=1}^{|\mathcal{V}|} \boldsymbol{\omega}_{i,j}^{(t)}$
    $\boldsymbol{\omega}_i^{(t+1)} \leftarrow \max(0, \boldsymbol{\omega}_i^{(t)} - \boldsymbol{\omega}_{\hat{\mathbf{y}}_t}^{(t)}) \quad \forall i \in \{1, \ldots, n\}$
    $\hat{\eta} \leftarrow \hat{\eta} + \sum_{j=1}^{|\mathcal{V}|} \boldsymbol{\omega}_{\hat{\mathbf{y}}_t, j}^{(t)}$
  end while
  $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_t)$
  return $\hat{\mathbf{y}}, \hat{\eta}$ ▷ Returns summary sentence indices and summary score.
end procedure

Figure 6: Simple sentence extraction algorithm given the sentence BOW representations $\omega_1, \omega_2, \ldots, \omega_n$, a word budget β, and a vector κ of sentence lengths (in words) as input.

reference abstract and 0 otherwise. The $\mathcal{L}_{ext}$ would then be augmented with an additional cross entropy loss for the word level predictions:

$$\mathcal{L}_{word}(z, \zeta; \boldsymbol{\theta}) = -\sum_{i=1}^{m} \zeta_i \log z_i + (1 - \zeta_i) \log(1 - z_i).$$

3.3.2 Proposed Experiments

Our main experiment will be to demonstrate that the performance of this word level model matches or exceeds the performance of the sentence extractor models from the previous section. Even only matching performance would be useful, since the word level model would add interpretability.

Additionally, we will further demonstrate the utility of word level salience estimates by using them as plug-in replacements for TF-IDF weights or unigram probabilities in unsupervised methods like LEXRANK (Erkan and Radev, 2004) and SUMBASIC (Nenkova and Vanderwende, 2005), i.e. using word importance should lead to improvements in these sentence extraction algorithms.

We also propose adapting the word level model to the news MDS task. We will do this in two ways. The first is to run LEXRANK and SUMBASIC methods on the MDS data with our drop-in word salience scores. These methods have been successfully applied to MDS and we expect that they would benefit from supervised word importance scores.

We also propose a simple self attention-based modification to the word importance aggregation step to help adapt our margin learning framework to MDS. We plan to experiment with the following importance aggregation method. First, given the outputs of the contextual features $\mathbf{h}_i = \text{CONTEXT}(w_i)$, we compute a self attention matrix $\Lambda \in \mathbb{R}^{m \times m}$ where

$$\Lambda_{i,j} = \sigma(\mathbf{h}_i^T \mathbf{h}_j / \tau + b)$$

using sigmoidal attention (Kim et al., 2017) with a learned bias parameter b and a temperature parameter τ. Next we compute an attention weighted word importance score $\bar{z}_i$ for each word in the input using the following formula,

$$\bar{z}_i = \sum_{j=1}^{m} z_j \cdot \Lambda_{i,j}.$$

We then use the aggregated word importance scores $\bar{z}_i$ in place of their non-aggregated counterparts in the creating the BOW representations $\boldsymbol{\omega}_j$.

Our motivation is that by accumulating scores based on context similarity, words and topics that appear in multiple documents will accumulate the bulk of the word importance scores, giving an added boost to sentences that contain them. Implicitly, errant words from one document that are not on topic to the cluster will not contribute much to a sentences score, reducing the effective dimension of the BOW vectors and regularizing individual sentences to the document cluster's mean. Conroy et al. (2013) found that dimensionality reduction on the BOW representations improves summarizer performance in the MDS setting (but not as much on single document summarization).

Since we are training our models on single documents, we expect that running our pretrainined word scoring model on the individual documents from an MDS document cluster will result in minimal task-adaptation mismatch. The remaining bias and temperature parameters can easily be tuned on the small amount of MDS training data available.

Finally, we would perform a similar series of ablation experiments as in our sentence salience paper, where we remove various feature representations or scramble the document or sentence order, so that we might better understand what learning signals are most important in the data. These experiments could inform additional genre transfer experiments. E.g. we could train a word level model only on adjectives and adverbs in the news domain and see if that model transfered better to the Reddit stories than the full news model.

If our margin learning framework does not work out, we should still be able to gain some benefit from using additional word feature embeddings in our existing sentence extractor models. We could perform ablation experiments to see if richer word representations changed the focus of learning from the implicit structure features that were dominant previously. Additionally we could introduce a similar word level auxiliary objective to the sentence extractor models to provide additional interpretability, if not performance.

4 Faithful Text Generation

Sections 2 and 3 have focused on identifying the most important content for summary inclusion, while punting somewhat on summary generation – and for good reason, extractive summarization minimizes the burden of creating fluent text. While abstractive text generation certainly adds significant challenges to summary creation, its benefits are many: the ability to achieve tighter compression ratios for space constrained scenarios (Fan et al., 2017), the potential to target different reading levels (Margarido et al., 2008) or style (Shen et al., 2017), and more pragmatically, in an increasingly copy-protected web, abstractive generation may be the only legally viable option for content aggregation services (Kassam, 2014).

With this expressive power comes the danger that the generated text may misconstrue the source material. Trust in machine learning models is increasingly being recognized as an important factor in user adoption (Ribeiro et al., 2016), and mistakes of this kind will be a show stopper for downstream consumers of summarization (e.g. if an abstractive summarization model in the previous crisis-monitoring scenario erroneously attributes the location of a deadly earthquake).

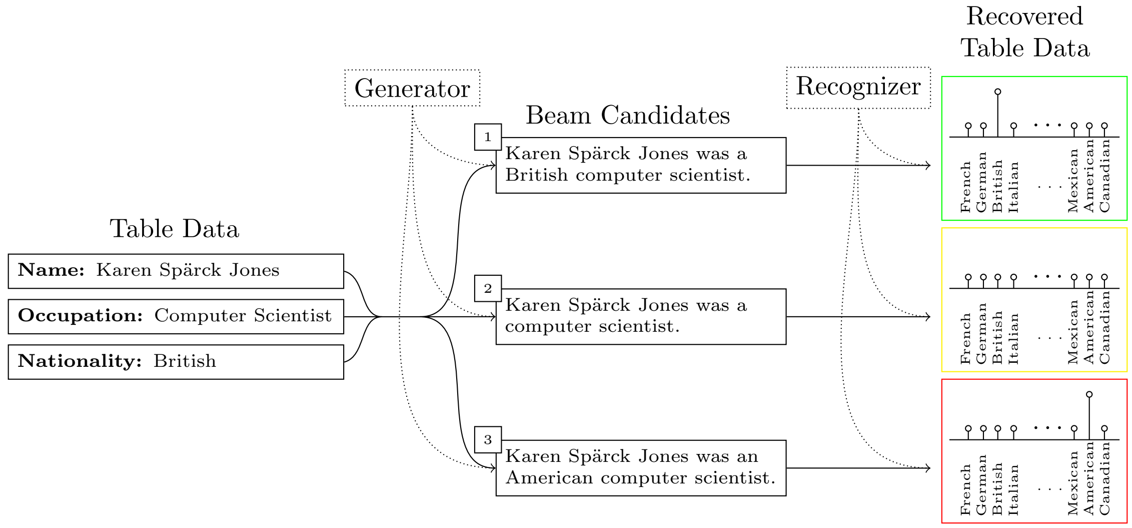

Figure 7: Example of faithful generation from structured data. The generator is responsible for producing a list (beam) of candidate utterances from the structured data. The recognizer reranks the beam candidates based on the plausibility of the recovered structured data.

As a potential solution, we propose modeling text generation as a two player game between a generator model and a recognizer model. First, we provide as input to the generator some evidence (e.g., raw text or table data). Conditioned on the evidence, the generator must produce a list of candidate utterances accurately describing that evidence. The recognizer scores the candidates based on its ability to reconstruct various pieces of the evidence. The generator receives supervision in the form of a reference utterance (i.e. standard sequence maximum likelihood training) and also from the recognizer scores across all candidate utterances. I.e., the generator learns to produce a variety of utterances that are simultaneously fluent but also lead the recognizer to recover the original evidence correctly. If we have an accurate recognizer, the generator should improve in its ability to render truthful descriptions of the evidence.

There are several motivations for this approach. First, RNNs typically used to implement text generators are impressive conditional language models and will generally learn to create fluent outputs. By adding a faithfulness component (recognizer derived losses) to the learning objective we can hopefully improve reliability of text generation for downstream users.

Second, by learning across multiple (beam search) candidates, we can encourage the model to explore syntactically and lexically diverse paraphrases that are still factually licensed by the evidence. We think this will be usefull when text generation is part of a pipeline that may want to consider various realizations to ensure fluency and coherency at a macro, e.g. document, level.

Furthermore, the recognizer is itself a learned model; as text classification models improve, so will the faithful generation framework. It does not require alignments between the evidence and the reference utterances; it uses the same evidence/utterance parallel data as the generator to train. The recognizer can also be used to derive confidence scores for individual utterances if this is needed in a downstream application.

Finally, we think this framework also allows for an alternative approach to controllable text generation. Currently, most controllable approaches to generation rely on appending feature embeddings to the decoder input (e.g. for length (Fan et al., 2017) or for syntactic structure (Colin and Gardent, 2018)). By specifying a desired output from the recognizer, we can train biased generators that adhere to soft constraints, similar to these other methods. What's more, the recognizer allows us to certify that individual outputs actually adhere to the desired constraints, something the input only control mechanisms do not do.

See Figure 7 for an example where we have table data as evidence (biographical data about name, nationality, and occupation), and we generate several plausible and implausible candidates that are evaluated by the recognizer. The first candidate is ranked highly because the recognizer can correctly infer that the nationality of Karen Spärck Jones is British (green box). The second candidate is possibly fine because it maximizes entropy over the choice of nationalies (yellow box), i.e. it makes no commitments either way and does not produce a non-true statement. The third beam candidate is clearly wrong as the recognizer infers that the nationality is American which is a false statement according to the true table data (red box). In some applications it may be OK to omit information, in which case the first and second utterances are acceptable and we only want to train the generator to avoid the third. In other cases, we may want to require that all input evidence has some corresponding description in the output, in which case, only the first utterance is acceptable. The faithful generation framework allows us to learn what the correct possible outputs are by specifying which recognizer distributions are acceptable.

In the remainder of this section, we cover the related work in this area before describing in more detail the faithful generation framework for data-to-text and text-to-text scenarios and our proposed experiments.

4.1 Related Work

The conceptual underpinnings of faithful generation trace back to two ideas in the NLP literature. The first is round-tripping in machine translation (Somers, 2005; Rapp, 2009), i.e. a good translation system should be able to translate a source language text to a target language text and then back to the source language with minimal corruption between the original source and the back-translated source. Andreas and Klein (2016) have recently carried this idea over to explaining structured prediction tasks, where, e.g., the structure is a formal plan for a robotic agent; a rational speaker generates the text description of the plan that is most likely to lead a rational listener to correctly re-interpret the original plan. Faithful generation is similar: the generator (speaker) encodes an input object (either text or data table) into an intermediate text representation, and the recognizer (listener) then tries to answer questions about the source using only the intermediate text. The main difference is that in faithful generation, we are interested not in a full reconstruction of the source from the intermediate representation, but rather, the utility of the intermediate representation on a downstream task, e.g. question answering.

The other conceptual precedent is that of discriminative reranking of n-best lists (Collins and Koo, 2005; Charniak and Johnson, 2005); a generative model produces the n most likely latent structures, e.g. parses, and a discriminative model, which does not have the burden of modeling the observations, rescores this list. This idea has been applied to text generation as well. Wen et al. (2015) and Novikova et al. (2017) use a sequence-to-sequence model with beam search to generate n-best texts from either a dialogue plan or data table respectively, and then use a separately trained classifier to downrank the beam candidates that do not correspond to the underlying structured object. We argue that this is insufficient since we might want to consider all beam candidates in a downstream task, e.g. a macro content planner stage. For this model to provide linguistically interesting realizations that are still factually true requires expanding the beam size which increases the computational and memory demands at test time. In our faithful generation framework, we directly discourage the generator from ever allowing an untrue statement to be kept in the beam. This is done using the recognizer directly as a learning signal and backpropagating through the beam search process (Wiseman and Rush, 2016).

Increasingly, summarization researchers are exploring REINFORCE-style policy gradient methods to optimize non-differentiable metrics like ROUGE (Paulus et al., 2017; Arumae and Liu, 2018; Kryściński et al., 2018; Narayan et al., 2018; Pasunuru and Bansal, 2018). The most related to our work is that of Arumae and Liu (2018) and Pasunuru and Bansal (2018). Arumae and Liu (2018) learn an extractive summarization model that maximizes performance of a question-answering model on cloze style questions created from the reference summaries. In our proposed method, the questions are generated from the source document and we use abstractively generated summaries as the input to the question answering model, a significantly harder task. Pasunuru and Bansal (2018) learn an abstractive summarization model that optimizes the likelihood that the generated summary is entailed by the ground-truth reference summary. The entailment likelihood is obtained from a model trained on the SNLI (Bowman et al., 2015) and MULTI-NLI (Williams et al., 2018) datasets. This entailment measure is somewhat orthogonal to the faithful generation objective, as our proposed approach directly evaluates the utility of the summary as a faithful proxy for the underlying document, not the reference, which is the ultimate goal of the summarization task.

Most reinforcement learning applied to abstractive summarization uses one Monte-Carlo sample from their policy distribution to estimate the expected reward. We propose to optimize the entire beam search so that all beam candidates are viable in downstream tasks. In this way, faithful generation also resembles minimum error rate training (MERT) (Och, 2003) and minimum Bayes-risk decoding (MBD) (Kumar and Byrne, 2004) in that we are reshaping a distribution of n-best beam candidates to optimize the expected value of an evaluation metric. We also plan on using reward shaping supervision from the recognizer model to localize reward signals (Mnih and Gregor, 2014) to specific spans of a candidate utterance that are most responsible for violating the input document. Localized reward shaping will help to penalize only the factually incorrect spans and preserve syntactic and structural choices that are independent of such entailment considerations.

Other notable approaches to improving the faithfulness of abstractive generation include Guo et al. (2018) who use a multi-task training objective to learn a shared decoder that can alternatively summarize a document, generate a question about a document, or generate a logically entailed text from a document. The latter task is relevant here, as the authors claim that this entailment generation objective encourages the decoder to produce only logically entailed summaries. This claim deserves further scrutiny as their human evaluation only asked about *relevance* and *fluency* where the relevance criteria included topical relevance and redundancy in addition to factual accuracy, confounding any interpretation of the generated summaries as being more faithful to the source document.

4.2 Data-to-Text Model

Our data-to-text model consists of evidence/utterance tuples $(x, y) \in \mathcal{D}$. The evidence $x = \{x_1, \ldots, x_m\}$ is a sequence of m categorical variables drawn from $\mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ where x_i is the observed value for the i-th field in the structured data, e.g. the i-th field could correspond to *occupation* with possible values in $\mathcal{X}_i = \{\text{ACCOUNTANT}, \text{ACTOR}, \ldots, \text{ZOOLOGIST}\}$. The utterance $y = \{y_1, \ldots, y_{|y|}\}$ is a sequence of $|y|$ tokens with each token drawn from a fixed vocabulary $\mathcal{V}$. The utterances correspond to natural language realizations of a subset of the evidence.

The first player in our game is the *generator* p, which can generate a list of k candidate utterances $y^{(1)}, \ldots, y^{(k)}$. In practice, p is a sequence-to-sequence model (Bahdanau et al., 2014) with parameters θ, and the k candidates are produced using beam search. The second player, called the *recognizer*, has a mapping $q_i : \mathcal{V}^* \to (0,1)^{|\mathcal{X}_i|}$ of utterances to probabilities over values for each field $i \in [\![m]\!]$. We say that an utterance y is **faithful** to the evidence x under a recognizer q, denoted $\textsc{Faithful}(y, x, q)$, if, for all $i \in [\![m]\!]$,

$$q_i(x_i|y) > \max_{x' \in \mathcal{X}_i \setminus \{x_i\}} q_i(x'|y) \quad \text{or} \quad H^{(i)}_{\max} - H_{q_i(\cdot|y)} < \epsilon$$

where $H^{(i)}_{\max}$ is the maximum entropy for the i-th field, i.e. the entropy of the uniform distribution $\log |\mathcal{X}_i|$, and $H_{q_i(\cdot|y)}$ is the entropy of q over the i-th field. In English, an utterance is faithful to the evidence if the recognizer can correctly predict the true evidence from the utterance (left statement) or, barring that, the recognizer cannot infer a value of the evidence with anything better than random chance (right statement).

We evaluate the degree to which the generator is faithful with the quantity

$$\mathcal{L}_\beta(x) = \mathbb{E}_{y \sim p(\cdot|x;\theta)} \mathbb{1}\{\textsc{Faithful}(y, x, q)\},$$

where $p(\cdot|x;\theta)$ is approximated using beam search.

We implement the recognizer as a collection of separate neural network classifiers q_i with parameters ϕ_i for each data field, and fit the parameters with the average log likelihood objective

$$\mathcal{L}_q(\phi) = \frac{1}{|\mathcal{D}|} \frac{1}{m} \sum_{x,y \in \mathcal{D}} \sum_{i=1}^{m} \log q_i(x_i|y;\phi_i).$$

After training, the recognizer is frozen and we do not update the parameters when fitting the generator.

The generator is pre-trained also using the standard maximum log likelihood objective:

$$\mathcal{L}_p(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \log p(y|x;\theta).$$

To make the generator faithful, we then maximize the joint objective

$$\frac{1}{|\mathcal{D}|} \sum_{x,y \in \mathcal{D}} \Big[\log p(y|x;\theta) + \mathbb{E}_{y \sim p(\cdot|x;\theta)} \mathbb{1}\{\textsc{Faithful}(y, x, q)\} \Big],$$

where the second term results in a REINFORCE style gradient update (Edunov et al., 2018).

We plan to validate the utility of this approach on two data-to-text tasks. The first is a biography generation task using biographical Wikipedia entries and structured data extracted from the corresponding Wikipedia page's info box (Lebret et al., 2016). The second data-to-text task is a restaurant description generation task, where the structured data consists of data about restaurants (types of food offered, location, etc.) (Novikova et al., 2017). Additionally, we will perform a human evaluation where we measure the accuracy of the beam candidates with respect to the table data.

4.3 Text-to-Text Model

For cases where the evidence is not structured data but text, e.g. summarization, we can modify the data-to-text model slightly to obtain a workable faithful training regime. Our data will

Input Text (x)	The BBC producer Oisin Tymon allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said . . . .
Reference Abstract (y)	Producer Oisin Tymon will not press charges against Jeremy Clarkson, his lawyer says.
Cloze Question ($\bar{x}$)	The BBC producer ■ ■ allegedly struck by Jeremy Clarkson will not press charges against the “Top Gear” host, his lawyer said.
Cloze Answer ($\bar{y}$)	Oisin Tymon

Figure 8: Example Summarization Cloze Question

now consist of 4-tuples $(x, y, \bar{x}, \bar{y}) \in \mathcal{D}$ where x is the input text to be summarized, y is the reference abstractive summary, and $\bar{x}, \bar{y}$ are cloze style questions and answers (Taylor, 1953) respectively. In a typical cloze question, a passage is given followed by a sentence with a missing word; one must provide the correct word to fill in the blank based on evidence from the passage. We can heuristically create cloze style questions for summarization by selecting input phrases/sentences with high similarity to the reference summary, and redacting random content words. See Figure 8 for an example from the CNN/DailyMail dataset.

We modify our recognizer definition to the following: an utterance y is **faithful** to the evidence x under a recognizer q, denoted $\textsc{Faithful}(y, x, q)$, if

$$q(\bar{y}|\bar{x}, y) > \max_{\bar{y}' \in \mathcal{V} \setminus \{\bar{y}\}} q(\bar{y}'|\bar{x}, y) \quad \text{or} \quad H_{\max} - H_{q(\cdot|\bar{x}, y)} < \epsilon$$

where the entropy terms are defined similarly to subsection 4.2 and the recognizer is now modified to map cloze question/passage tuples $\bar{x}, y$ to probabilities of a cloze answer $\bar{y}$. In practice multiple cloze question/answer pairs will be created for each input/summary pair. Training of the faithful generator will proceed similarly to the process outlined in subsection 4.2.

We plan to perform faithful summary generation first using the TL;DR dataset (Völske et al., 2017) which contains over 2 million Reddit comments with summarizations. Since the comments being summarized are shorted than most news articles, we think this will be a more tractable starting point for validating the utility of the cloze task. Following this, we would like to apply this technique to the CNN/DailyMail, NYT, and Newsroom datasets (Hermann et al., 2015; Sandhaus, 2008; Grusky et al., 2018) since these datasets are large enough to train abstractive summarizers. As in the data-to-text experiments, we will also include a human evaluation of the beam candidates to ensure that the training does indeed produce a distribution of faithful outputs.

5 Research Plan

Table 8 shows our planned timeline to complete the remaining research. We plan to participate in the TL;DR text generation challenge (Syed et al., 2018) with our faithful generation method. The TL;DR deadline is April 1st. Additionally, we would like to publish this work at ACL and so we will target the March 4th submission deadline. Work on the word importance estimation is more evergreen and so we will begin this research in the spring and hopefully finish in the summer, possibly submitting to EMNLP or NAACL.

Task	**Date**
Stream Summarization (SAP)	Completed
Stream Summarization (L2S)	Completed
Deep Learning Sentence Salience	Completed
Data-to-Text Implementation	December 2019
Text-to-Text Implementation	January 2019
Finetuning/Auto and Human evaluation (ACL)	February 2019 - March 2019
Word Importance (SDS)	April 2019 - May 2019
Word Importance (MDS and Genre Adaptation)	July 2019
Word Importance (Abstractive Summarization)	June 2019
Write Thesis	August 2019 - February 2020
¡Defend!	March 2020

Table 8: Research Plan

6 Conclusion

In this thesis we have proposed to tackle two central problems for summarization: selecting salient information for summary inclusion and generating text that is grounded in the underlying structured or text data. Our experiments on the salience problem span many different summarization scenarios, including the very challenging stream summarization task. We also show how to incorporate salience estimation into two very different paradigms, exemplar based clustering and learning-to-search. We also contribute analysis as to the behavior of several deep learning models of sentence salience, and, based on these experiments, propose a novel word importance estimation model, with applications to single and multi-document summarization. Moreover, we plan to explore genre adaption and integration with abstractive summarization to further demonstrate the value of our approach. We hope that these sections of the thesis form a flexible collection of strategies useful to a broad range of researchers in summarization.

This work is diminished, however, if we do not have robust generation algorithms that do not hallucinate information. We believe the final chapter on faithful generation will prove a useful paradigm for dealing with errorful generation outputs. By experimenting on both data-to-text and text-to-text scenarios we hope to show the utility of our approach across a variety of text generation scenarios. Additionally, we think faithful generation will prove to be a useful alternative method of controllable generation that adds more guarantees that the generated output conforms to the desired constraints.

In the coming year we hope to provide evidence across a range of tasks that our paradigm makes substantial progress on these challenges in summarization and generation, culminating in the successful defense of this thesis.

7 References

Jacob Andreas and Dan Klein. Reasoning about pragmatics with neural listeners and speakers. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, 2016.

Kristjan Arumae and Fei Liu. Reinforced extractive summarization with question-focused rewards. *arXiv preprint arXiv:1805.10392*, 2018.

Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. Trec 2013 temporal summarization track overview. 2014.

Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. Trec 2014 temporal summarization track overview. 2015.

Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. Trec 2015 temporal summarization track overview. 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490, 2011.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

Ronald Brandow, Karl Mitze, and Lisa Rau. Automatic condensation of electronic publications by sentence selection. In Jan Fagerberg, David C. Mowery, and Richard R. Nelson, editors, *Advances in Automatic Text Summarization*, chapter 19, pages 293–303. MIT Press, Oxford, 1999. ISBN 9780262133593. URL `https://books.google.com/books?id=YtUZQaKDmzEC`.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 2058–2066. JMLR. org, 2015.

Eugene Charniak and Mark Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 173–180. Association for Computational Linguistics, 2005.

Shouyuan Chen, Yuanming Yu, Chong Long, Feng Jin, Lijing Qin, Minlie Huang, and Xiaoyan Zhu. Tsinghua university at the summarization track of tac 2008. In *TAC*. Citeseer, 2008.

Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–494, 2016.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

Emilie Colin and Claire Gardent. Generating syntactic paraphrases. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 937–943, 2018.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1), 2005.

John Conroy, Sashka T Davis, Jeff Kubina, Yi-Kai Liu, Dianne P Oleary, and Judith D Schlesinger. Multilingual summarization: Dimensionality reduction and a step towards optimal term coverage. In *Proceedings of the MultiLing 2013 Workshop on Multilingual Multi-document Summarization*, pages 55–63, 2013.

John M Conroy and Dianne P O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM, 2001.

John M Conroy, Judith D Schlesinger, and Dianne P O'Leary. Nouveau-rouge: A novelty metric for update summarization. *Computational Linguistics*, 37(1):1–8, 2011.

Hoa Trang Dang and Karolina Owczarzak. Overview of the tac 2008 update summarization task.

Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1998–2008, 2016.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, et al. Classical structured prediction losses for sequence to sequence 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)*, volume 1, pages 355–364, 2018.

Günes Erkan. Using biased random walks for focused summarization. *Ann Arbor*, 1001: 48109–2121, 2006.

Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

Angela Fan, David Grangier, and Michael Auli. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*, 2017.

John R Frank, Max Kleiman-Weiner, Daniel A Roberts, Feng Niu, Ce Zhang, Christopher Ré, and Ian Soboroff. Building an entity-centric stream filtering test collection for trec 2012. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE, 2012.

Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, 2018.

Max Grusky, Mor Naaman, and Yoav Artzi. 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)*, volume 1, pages 708–719, 2018.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Soft layer-specific multi-task summarization with entailment and question generation. *arXiv preprint arXiv:1805.11004*, 2018.

Qi Guo, Fernando Diaz, and Elad Yom-Tov. Updating users about time critical events. In *Proceedings of the 35th European conference on Advances in Information Retrieval*, pages 483–494. Springer-Verlag, 2013.

Weiwei Guo and Mona Diab. A simple unsupervised latent semantics based approach for sentence similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 586–590. Association for Computational Linguistics, 2012.

Z.S. Harris. *String Analysis of Sentence Structure*. Papers on formal linguistics. Mouton, 1962. URL https://books.google.com/books?id=4dssAAAAMAAJ.

Ruifang He, Yang Liu, Bing Qin, Ting Liu, and Sheng Li. Hitir's update summary at tac 2008: Extractive content selection for language independence. In *TAC*, 2008.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

Tsutomu Hirao, Yutaka Sasaki, Hideki Isozaki, and Eisaku Maeda. Ntts text summarization system for duc-2002. Citeseer, 2002.

Kai Hong and Ani Nenkova. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721, 2014.

Ashifa Kassam. Google news says 'adiós' to spain in row over publishing fees. *The Guardian*, 2014. URL https://www.theguardian.com/world/2014/dec/16/google-news-spain-publishing-fees-internet.

Chris Kedzie, Kathleen McKeown, and Fernando Diaz. Predicting salient updates for disaster summarization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1608–1617, 2015.

Chris Kedzie, Fernando Diaz, and Kathleen McKeown. Real-time web scale event summarization using sequential decision making. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3754–3760. AAAI Press, 2016. ISBN 978-1-57735-770-4. URL http://dl.acm.org/citation.cfm?id=3061053.3061144.

Chris Kedzie, Kathleen McKeown, and Hal Daumé III. Content selection in deep learning models of summarization. In *EMNLP*, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R Kneser and H Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE, 1995.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. Improving abstraction in text summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1808–1817, 2018.

Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, 2004.

Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73. ACM, 1995.

Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, 2016.

C-Y Lin. Rouge: A package for automatic evaluation of summaries. In *Proc. of Workshop on Text Summarization Branches Out, Post Conference Workshop of ACL 2004*, 2004.

Chin-Yew Lin and Eduard Hovy. The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 495–501. Association for Computational Linguistics, 2000.

Yong Liu, Xiaolei Wang, Jin Zhang, and Hongbo Xu. Personalized pagerank based multi-document summarization. In *Semantic Computing and Systems, 2008. WSCS'08. IEEE International Workshop on*, pages 169–173. IEEE, 2008.

Annie Louis. A bayesian method to incorporate background knowledge during automatic text summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 333–338, 2014.

Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

Paulo RA Margarido, Thiago AS Pardo, Gabriel M Antonio, Vinícius B Fuentes, Rachel Aires, Sandra M Aluísio, and Renata PM Fortes. Automatic summarization for text simplification: Evaluating text understanding by poor readers. In *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, pages 310–315. ACM, 2008.

André FT Martins and Noah A Smith. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 1–9. Association for Computational Linguistics, 2009.

Richard McCreadie, Craig Macdonald, and Iadh Ounis. Incremental update summarization: Adaptive sentence selection based on prevalence and novelty. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 301–310. ACM, 2014.

Richard McCreadie, Saul Vargas, Craig MacDonald, Iadh Ounis, Stuart Mackie, Jarana Manotumruksa, and Graham McDonald. University of glasgow at trec 2015: Experiments with terrier in contextual suggestion, temporal summarisation and dynamic domain tracks. In *TREC*, 2015.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799, 2014.

Saif Mohammad, Bonnie J Dorr, Melissa Egan, Nitin Madnani, David M Zajic, and Jimmy J Lin. Multiple alternative sentence compressions and word-pair antonymy for automatic text summarization and recognizing textual entailment. 2008.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *arXiv preprint arXiv:1611.04230*, 2016a.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*, 2016b.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 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)*, volume 1, pages 1747–1759, 2018.

Ani Nenkova. Automatic text summarization of newswire: lessons learned from the document understanding conference. In *Proceedings of the 20th national conference on Artificial intelligence-Volume 3*, pages 1436–1441. AAAI Press, 2005.

Ani Nenkova and Lucy Vanderwende. The impact of frequency on summarization. In *Microsoft Research*. Citeseer, 2005.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, 2017.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

Miles Osborne. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 1–8. Association for Computational Linguistics, 2002.

Ramakanth Pasunuru and Mohit Bansal. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 646–653, 2018.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

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

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

Reinhard Rapp. The back-translation score: automatic mt evaluation at the sentence level without reference translations. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 133–136. Association for Computational Linguistics, 2009.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

Ahsan Raza, Devin Rotondo, and Charles Clarke. Waterlooclarke: Trec 2015 temporal summarization track. In *TREC*, 2015. doi: http://trec.nist.gov/pubs/trec24/papers/WaterlooClarke-TS.pdf.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

Ralph Rose. The relative contribution of syntactic and semantic prominence to the salience of discourse entities. *Unpublished PhD, Northwestern University*, 2005.

Evan Sandhaus. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752, 2008.

Imran Sheikh, Irina Illina, Dominique Fohr, and Georges Linares. Learning word importance with the neural bag-of-words model. In *ACL, Representation Learning for NLP (Repl4NLP) workshop*, 2016.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2862–2867. Morgan Kaufmann Publishers Inc., 2007.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6830–6841. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7259-style-transfer-from-non-parallel-text-by-cross-alignment.pdf`.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233. Association for Computational Linguistics, 2012.

Harold Somers. Round-trip translation: What is it good for? In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 127–133, 2005.

Kate Starbird and Leysia Palen. Working and sustaining the virtual disaster desk. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 491–502. ACM, 2013.

Shahbaz Syed, Michael Völske, Martin Potthast, Nedim Lipka, Benno Stein, and Hinrich Schütze. Task proposal: The tl; dr challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 318–321, 2018.

Wilson L Taylor. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433, 1953.

S Teufel. Sentence extraction as a classification task. In *Proc. of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 58–65, 1997.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, 2017.

Tsung-Hsien Wen, Milica Gašic, Dongho Kim, Nikola Mrkšic, Pei-Hao Su, David Vandyke, and Steve Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 275, 2015.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1112–1122, 2018.

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016.

Tan Xu, Douglas W Oard, and Paul McNamee. Hltcoe at trec 2013: Temporal summarization. In *TREC*, 2013.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. Multi-document summarization by maximizing informative content-words. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1776–1782. Morgan Kaufmann Publishers Inc., 2007.

Victor H Yngve. Syntax and the problem of multiple meaning. *Machine translation of language*, 1955.

Jin Zhang, Xueqi Cheng, Hongbo Xu, Xiaolei Wang, and Yiling Zeng. Ictcas's ictgrasper at tac 2008: Summarizing dynamic information with signature terms based content filtering. Citeseer, 2008.